chroGPS 2.0: Visualization, functional analysis and comparison of epigenome maps.

Oscar Reina *and David Rossell *

1 Citation

For citation of the chroGPS package, please use the following reference: [Font-Burgada et al., 2013].

2 Introduction

The chroGPS package provides tools to generate and analyze intuitive maps to visualize and compare the association between genetic elements, with emphasis on epigenetics. The approach is based on Multi-Dimensional Scaling, hierarchival clustering and Procrustes. 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 [Font-Burgada et al., 2013]. Major changes were introduced in the 2.0 version to account for differential analysis of the generated epigenome maps, as well as additional functions for manipulating genomic interval objects and to use our epigenetic overlap metrics to help in the selection of candidate epigenetic marks for de-novo experiments.

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.

3 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

GRangesList object of length 20:
$`ASH1-Q4177.S2`
```

*Bioinformatics & Biostatistics Unit, IRB Barcelona

```
GRanges object with 1813 ranges and 2 metadata columns:
         seqnames            ranges strand |             score
            <Rle>         <IRanges>  <Rle> |         <numeric>
     [1]    chr2L       84702-87247      * | 0.803803527114604
     [2]    chr2L     124999-129257      * |  1.04668069964713
     [3]    chr2L     140942-142153      * |  0.42272847440403
     [4]    chr2L     158853-159885      * |  1.63201617502988
     [5]    chr2L     479317-485541      * |  1.12086072790696
     ...      ...               ...    ... .               ...
  [1809]     chrX 22248009-22251087      * |   1.0120068501168
  [1810]     chrX 22251661-22253325      * |   1.1529662196046
  [1811]     chrX 22253449-22257063      * |  1.35826010289004
  [1812]     chrX 22409811-22412134      * |  1.73872309250348
  [1813]  chrXHet       77007-81264      * |  2.02974041804889
                                         ID
                                <character>
     [1]  ASH1_Q4177_S2.enriched_region_741
     [2]  ASH1_Q4177_S2.enriched_region_742
     [3]  ASH1_Q4177_S2.enriched_region_743
     [4]  ASH1_Q4177_S2.enriched_region_744
     [5]  ASH1_Q4177_S2.enriched_region_745
     ...                               ...
  [1809] ASH1_Q4177_S2.enriched_region_1546
  [1810] ASH1_Q4177_S2.enriched_region_1547
  [1811] ASH1_Q4177_S2.enriched_region_1548
  [1812] ASH1_Q4177_S2.enriched_region_1549
  [1813] ASH1_Q4177_S2.enriched_region_1813
  -------
  seqinfo: 13 sequences from an unspecified genome

...
<19 more elements>
```

s2 is a `GRangesList` object storing the binding sites for 20 Drosophila melanogaster S2-DRSC sample proteins. Data was retrieved from the modEncode website (www.modencode.org) and belongs to the public subset of the Release 29.1 dataset. GFF files were downloaded, read and formatted into individual GRanges objects, stored later into a `GRangesList` (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.

3.1 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 1. 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 1. 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)
```

3.2 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
```

```
GRangesList object of length 4:
$`GSM480156_dm3-S2-H3K4me3.bed.rd`
GRanges object with 7406 ranges and 2 metadata columns:
         seqnames        ranges strand | values.counts
            <Rle>     <IRanges>  <Rle> |     <integer>
     [1]    chr2L   72856-72925      * |            48
     [2]    chr2L   72960-72990      * |            34
     [3]    chr2L   73487-73555      * |            46
     [4]    chr2L   73574-73679      * |           144
     [5]    chr2L   73682-73842      * |           182
     ...      ...           ...    ... .           ...
  [7402]  chrXHet 191720-191749      * |            23
  [7403]  chrXHet 194583-194648      * |            51
  [7404]  chrXHet 194668-194730      * |            94
  [7405]  chrXHet 194734-194787      * |            55
  [7406]  chrXHet 194883-194927      * |            41
```

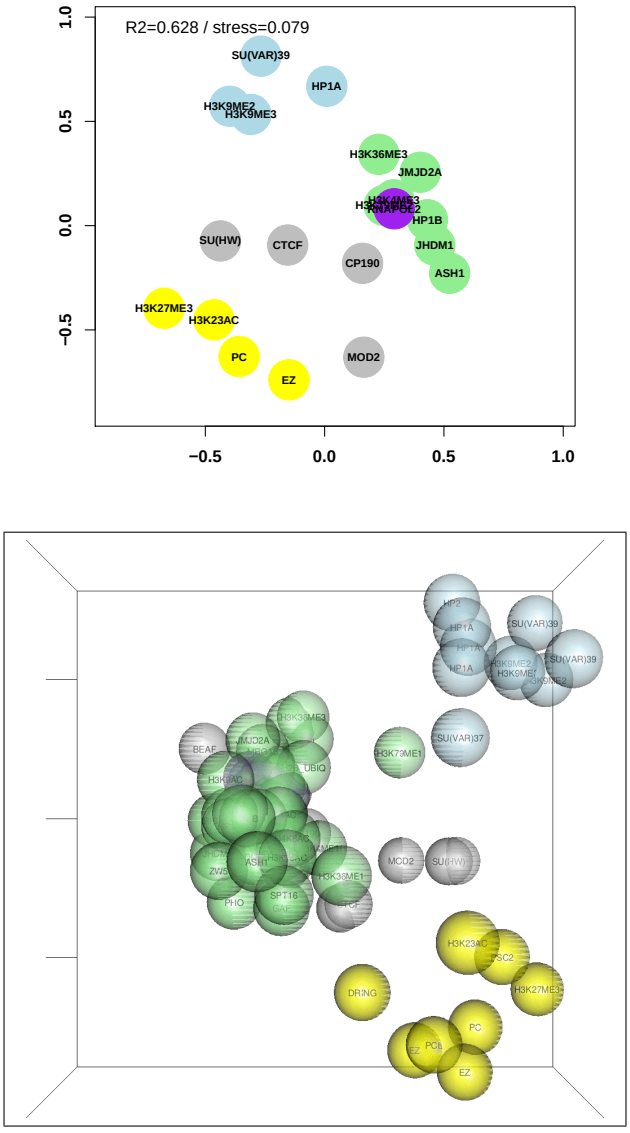

Figure 1: 2D map from the 20 S2 epigenetic factors and example 3D map with 76 S2 factors. Factors with more similar binding site distribution appear closer.

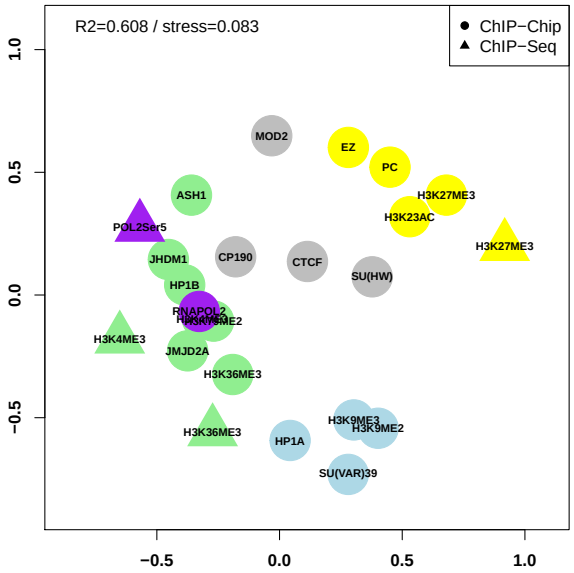

Figure 2: S2 ChIP-chip and ChIP-Seq data, raw integration (no adjustment).

```
              values.pvalue
                  <numeric>
    [1] 1.31963353957947e-07
    [2]  0.00354225608299729
    [3] 7.38655420659492e-07
    [4] 8.99177242511397e-71
    [5] 3.9046265654162e-105
    ...                  ...
  [7402]   0.0333937719058131
  [7403] 1.84180539954291e-13
  [7404] 1.21724769000859e-43
  [7405] 9.93039693670726e-16
  [7406] 2.01987740433528e-08
  -------
  seqinfo: 12 sequences from an unspecified genome

...
<3 more elements>
```

```
> # d2 <- distGPS(c(reduce(s2),reduce(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 2 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))
```

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).

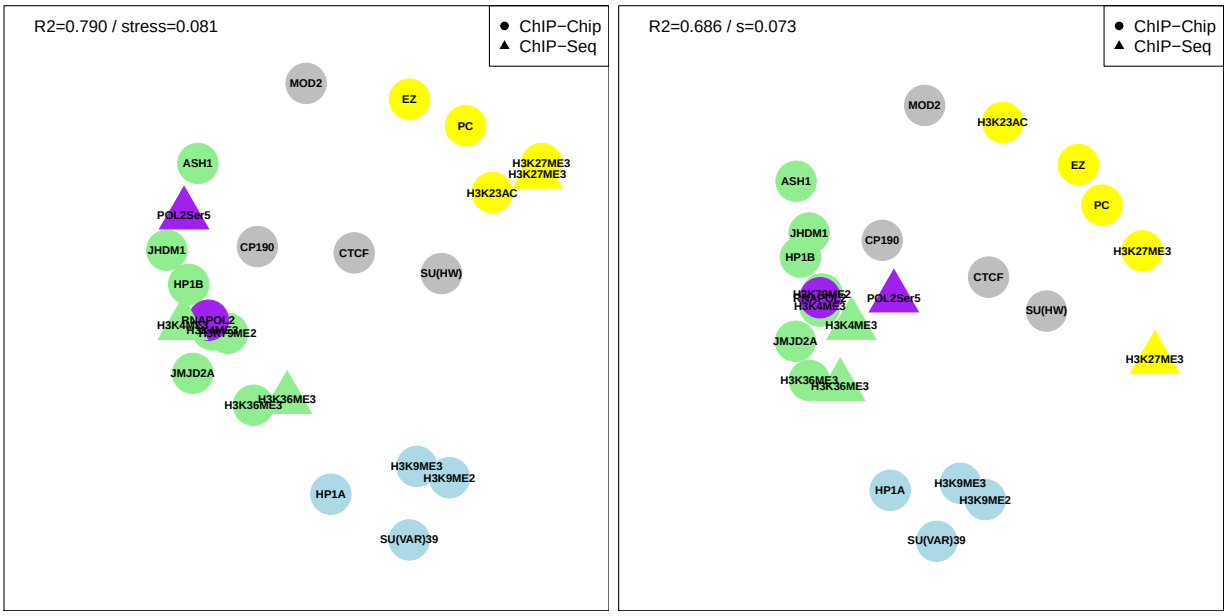

Figure 3: S2 ChIP-chip and ChIP-Seq data. Left: Procrustes adjustment. Right: Peak Width Adjustment.

```
> s2.pAdj <- adjustPeaks(c(reduce(s2),reduce(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 15 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.

4 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).

4.1 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 [Zhu et al., 2010]. 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 [Venables and Ripley, 2002].

```
> 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)))
```

4.2 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'))
```

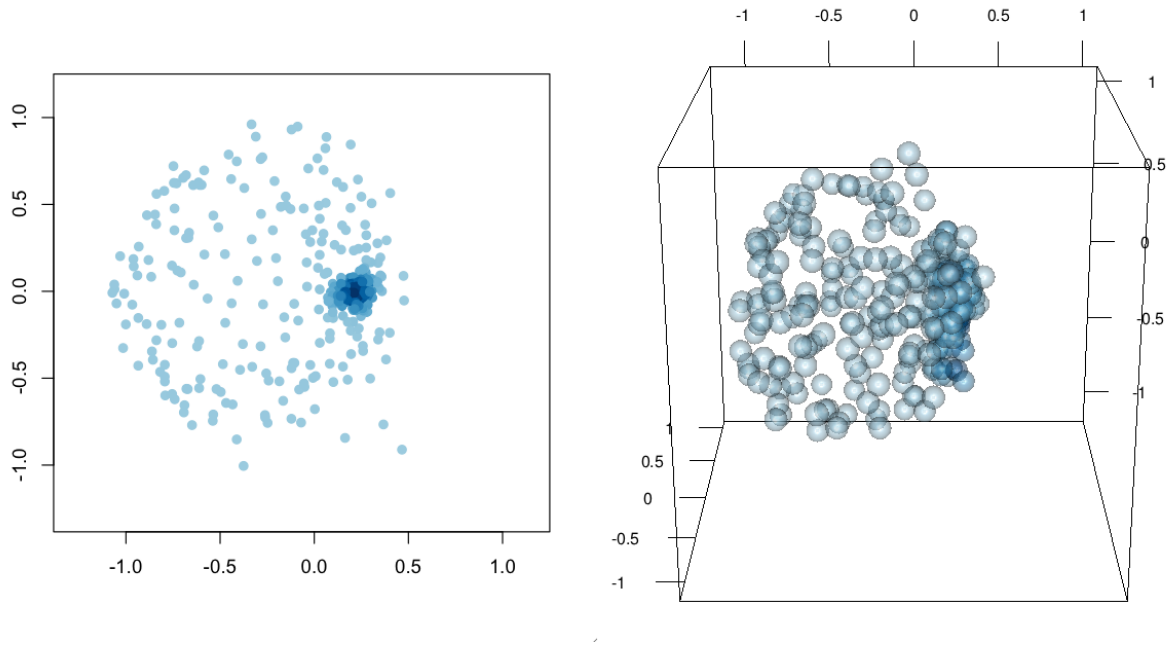

Figure 4: 2 and 3-dimensional chroGPS^{genes}. Genes with more similar epigenetic marks (binding site patterns) appear closer.

```
   user  system elapsed
  5.345   0.042   5.485
```

```
> 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 [Font-Burgada et al., 2013] 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
  2.238   0.042   2.302
```

```
> mds3
```

```
Object of class MDS approximating distances between 466 objects
R-squared= 0.8134 Stress= 0.1309
```

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

```
Sampling  100 elements...
   Correl   Step size
0.8100498
0.8330063 0.03735367
0.8379277 0.01454418
```

```
0.8386038 0.01364894
   user  system elapsed
  0.523   0.069   0.599
```

```
> mds4
```

```
Object of class MDS approximating distances between 466 objects
R-squared= 0.8469 Stress= 0.1212
```

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.

4.3 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 5

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

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  2.192   4.518   8.934   7.917  10.573  13.258      47
```

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

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  2.136   3.987   8.343   7.454  10.426  13.258      31
```

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

4.4 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

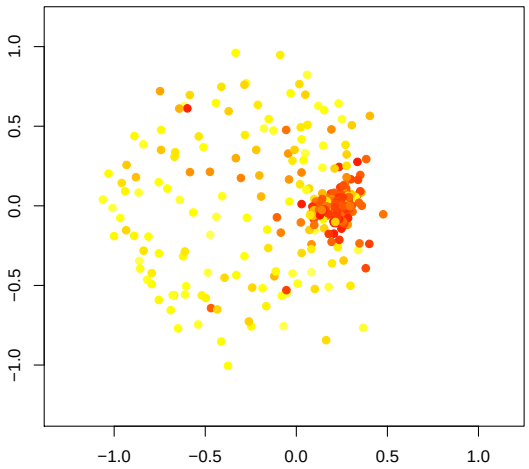

Figure 5: 2-dimensional MDS plot with chroGPS^{genes} map and gene expression information.

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 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 6. 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.

4.5 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)
```

5

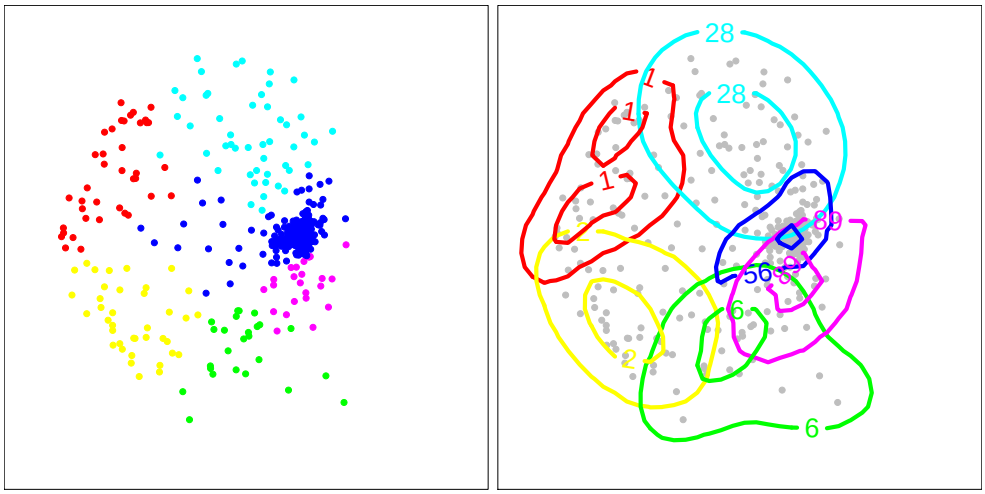

Figure 6: 2-dimensional MDS plot with chroGPS^{genes} map and cluster identities indicated by point colors (left) and probabilistic contours drawn at 50 and 95 percent (right).

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.

4.6 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 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 7 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()
```

4.7 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 9 (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).

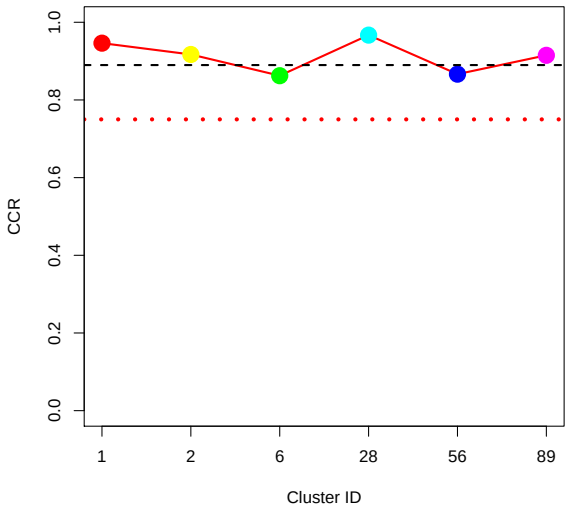

Figure 7: Per-cluster (dots and continuous line) and global (dashed line) Correct Classification Rate. Red pointed line indicates an arbitrary threshold of 0.75 CCR. Higher values indicate more robust clusters which are better separated in space.

```
> 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 9 (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)
> 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')
```

4.8 Merging overlapping clusters

As discussed in Section 4.6, 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

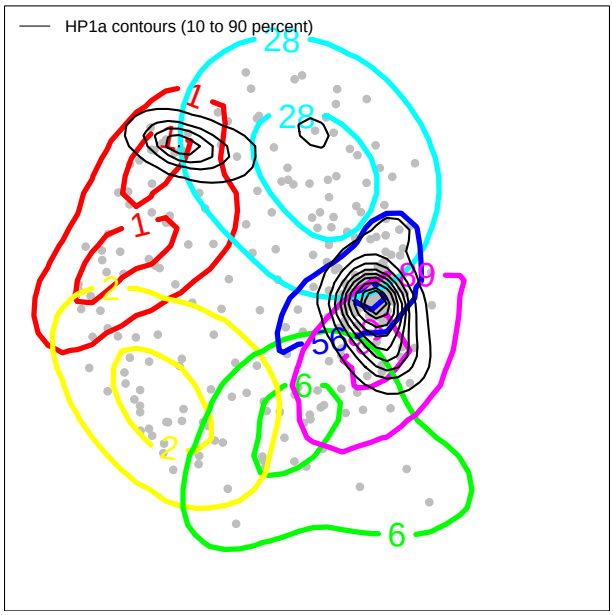

Figure 8: chroGPS^{genes} map with cluster contours at 50 and 95 percent the 5 clusters presented above. In black, probability contour for HP1a factor.

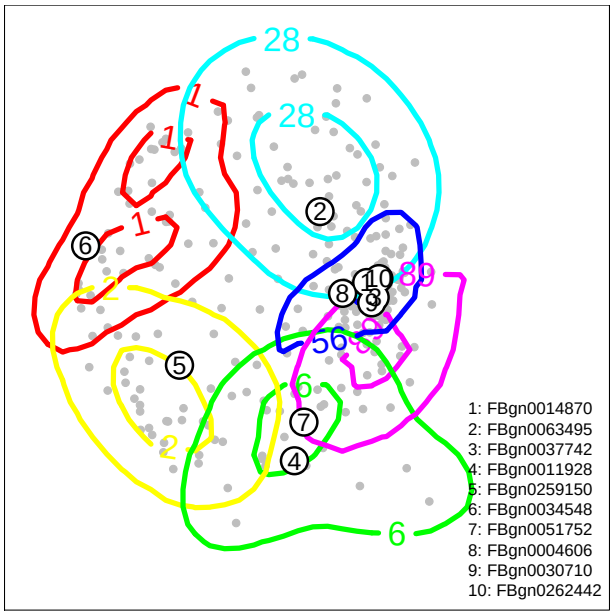

Figure 9: chroGPS^{genes} map with cluster contours at 50 and 95 percent the 5 clusters presented above. Left: In black, probability contour for HP1a factor. Right: random geneset located on the chroGPS^{genes} map.

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
```

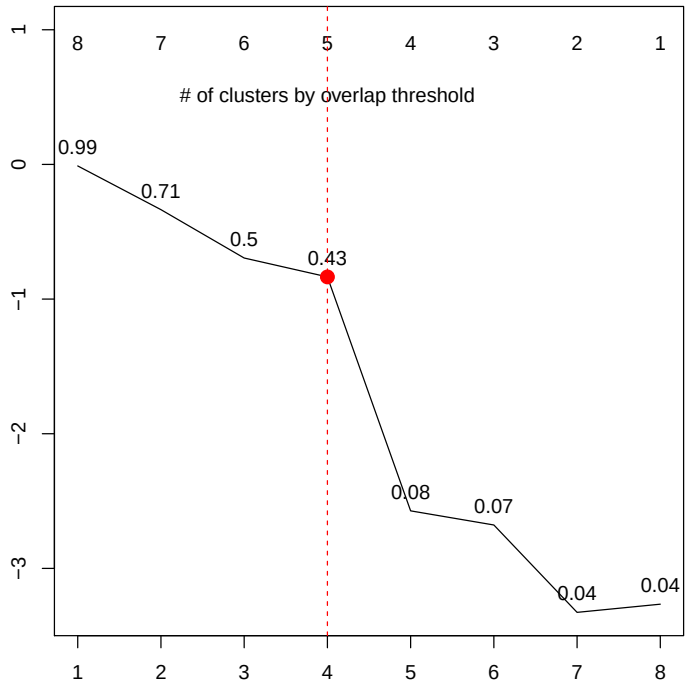

Figure 10: Overview of maximum cluster overlap observed in each merging step. Merging stops at 5 clusters, when the next two clusters to merge show an overlap differing significantly in mean to those from previous steps.

We plot the cluster contours before and after merging (Figure 11). 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 10. After merging all cluster-specific CCR values were roughly ≥ 0.9. The code required to produce Figure 11 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 (12)

```
> 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')
```

4.9 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 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 11.

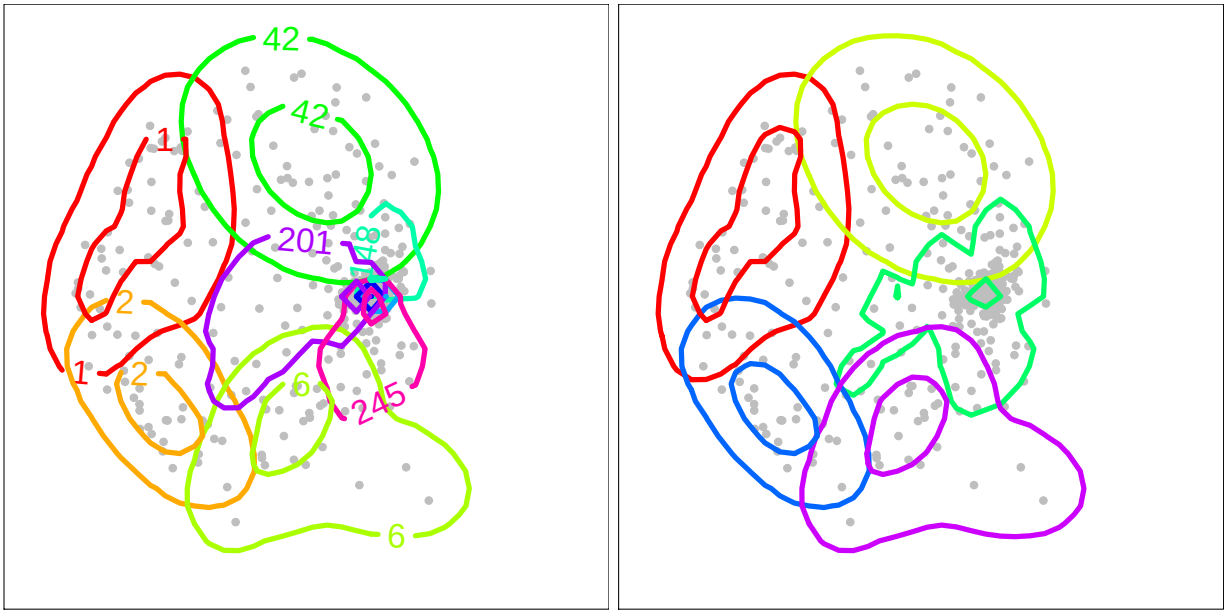

Figure 11: chroGPS^{genes} map with clusters at between-cluster distance of 0.2, and cluster density contours at 50 and 95 percent. Left: Unmerged. Right: Merged.

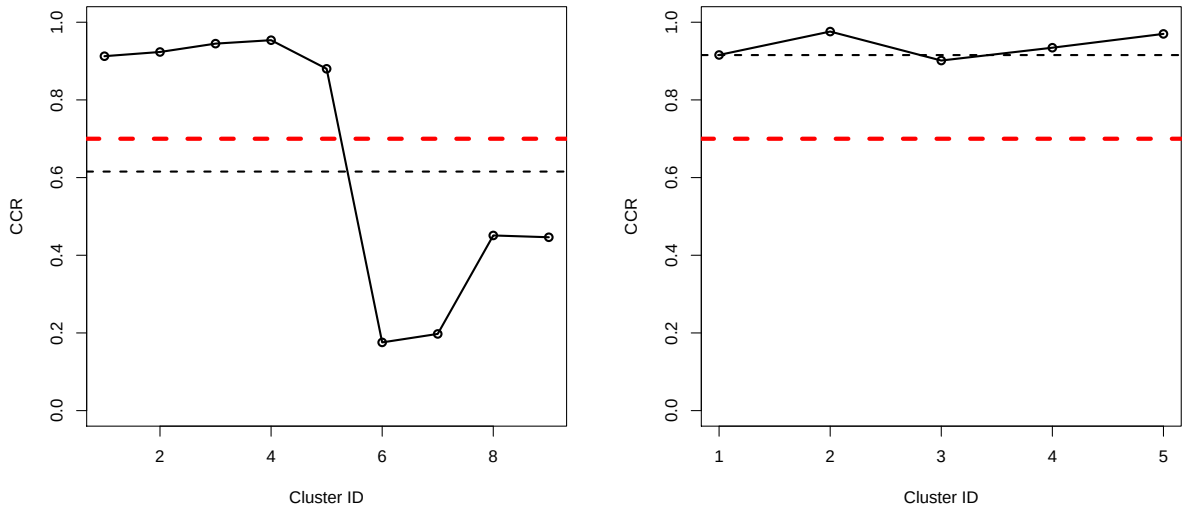

Figure 12: Per-cluster (dots and continuous line) and global (dashed line) mis-classification rate for the clusters shown in Figure 11. Red dashed line indicates an arbitrary threshold of 0.7 CCR. Left: Unmerged. Right: Merged

```
> 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')
```

5 Dealing with epigenetic replicates

When dealing with ChIP-chip and ChIP-Seq experiments, it is normal to encounter different replicates for the same epigenetic factors, which respond to the use of different antibodies or experimental procedures. We offer a function, called mergeReplicates, which can be used to join experimental or technical replicates at epigenetic factor level (that is, binding sites), as well as when having them in a genes x factors contingency table. The function allows for several criteria in order to merge replicates from the same factor.

```
> library(caTools)
> library(gplots)
> library(chroGPS)
> data(s2)
> data(bg3)
> s2
```

```
GRangesList object of length 20:
$`ASH1-Q4177.S2`
GRanges object with 1813 ranges and 2 metadata columns:
         seqnames            ranges strand |             score
            <Rle>         <IRanges>  <Rle> |         <numeric>
     [1]     chr2L       84702-87247      * | 0.803803527114604
     [2]     chr2L     124999-129257      * |  1.04668069964713
     [3]     chr2L     140942-142153      * |  0.42272847440403
     [4]     chr2L     158853-159885      * |  1.63201617502988
     [5]     chr2L     479317-485541      * |  1.12086072790696
     ...       ...               ...    ... .               ...
  [1809]      chrX 22248009-22251087      * |   1.0120068501168
  [1810]      chrX 22251661-22253325      * |   1.1529662196046
  [1811]      chrX 22253449-22257063      * |  1.35826010289004
  [1812]      chrX 22409811-22412134      * |  1.73872309250348
  [1813]   chrXHet       77007-81264      * |  2.02974041804889
                                       ID
                                <character>
     [1]  ASH1_Q4177_S2.enriched_region_741
     [2]  ASH1_Q4177_S2.enriched_region_742
     [3]  ASH1_Q4177_S2.enriched_region_743
     [4]  ASH1_Q4177_S2.enriched_region_744
     [5]  ASH1_Q4177_S2.enriched_region_745
     ...                               ...
  [1809] ASH1_Q4177_S2.enriched_region_1546
  [1810] ASH1_Q4177_S2.enriched_region_1547
  [1811] ASH1_Q4177_S2.enriched_region_1548
  [1812] ASH1_Q4177_S2.enriched_region_1549
  [1813] ASH1_Q4177_S2.enriched_region_1813
  -------
  seqinfo: 13 sequences from an unspecified genome

...
<19 more elements>
```

```
> bg3
```

```
GRangesList object of length 20:
$`ASH1-Q4177.BG3`
```

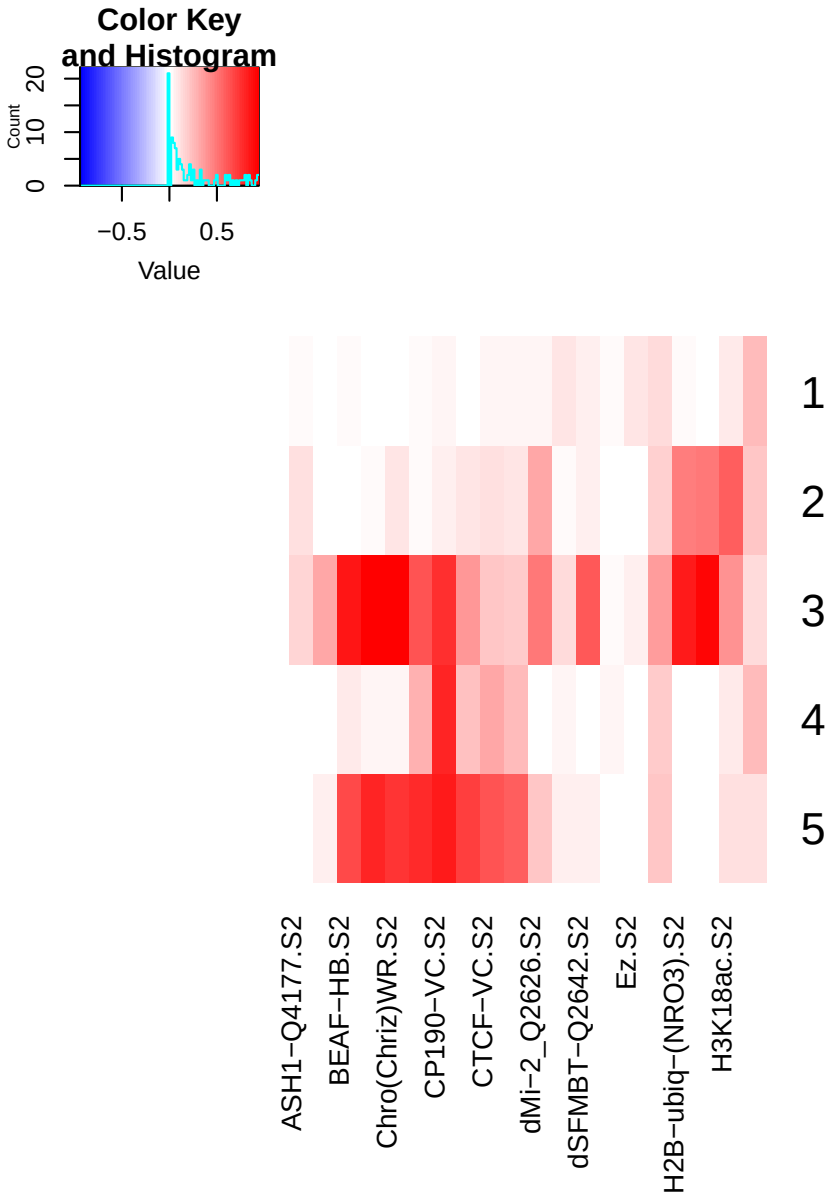

Figure 13: chroGPS^{genes} profile heatmap of the 9 unmerged clusters presented at Figure 11 after unsupervised merging of overlapping clusters (showing 20 first factors for visualization purposes). Merged clusters get concatenated names from the original clusters.

```
GRanges object with 2410 ranges and 2 metadata columns:
         seqnames            ranges strand |             score
            <Rle>         <IRanges>  <Rle> |         <numeric>
     [1]    chr2L     119812-121788      * |  0.64905422771079
     [2]    chr2L     125666-126674      * | 0.745198215686335
     [3]    chr2L     126896-130113      * |  1.05908122980415
     [4]    chr2L     132063-135854      * | 0.667792943275969
     [5]    chr2L     136096-137428      * | 0.551935649474012
     ...      ...               ...    ... .               ...
  [2406]     chrX 22246321-22247969      * | 0.722123189872635
  [2407]     chrX 22248204-22251087      * | 0.997457192239224
  [2408]     chrX 22251661-22253325      * |  1.16417178389549
  [2409]     chrX 22253449-22257285      * |  1.63252703592798
  [2410]     chrX 22261488-22262681      * | 0.453297135989299
                                          ID
                                 <character>
     [1] ASH1_Q4177_BG3.enriched_region_1028
     [2] ASH1_Q4177_BG3.enriched_region_1029
     [3] ASH1_Q4177_BG3.enriched_region_1030
     [4] ASH1_Q4177_BG3.enriched_region_1031
     [5] ASH1_Q4177_BG3.enriched_region_1032
     ...                                 ...
  [2406] ASH1_Q4177_BG3.enriched_region_2043
  [2407] ASH1_Q4177_BG3.enriched_region_2044
  [2408] ASH1_Q4177_BG3.enriched_region_2045
  [2409] ASH1_Q4177_BG3.enriched_region_2046
  [2410] ASH1_Q4177_BG3.enriched_region_2047
  -------
  seqinfo: 13 sequences from an unspecified genome

...
<19 more elements>
```

```
> # Unify replicates
> mnames <- sort(unique(intersect(s2names$Factor,bg3names$Factor)))
> sel <- s2names$Factor %in% mnames
> s2.repset <- mergeReplicates(s2[sel],id=s2names$Factor[sel],mergeBy='any')
> sel <- bg3names$Factor %in% mnames
> bg3.repset <- mergeReplicates(bg3[sel],id=bg3names$Factor[sel],mergeBy='any')
> # Show
> names(s2.repset)
 [1] "ASH1"      "CP190"     "CTCF"      "EZ"        "H3K23AC"
 [6] "H3K27ME3"  "H3K36ME3"  "H3K4ME3"   "H3K79ME2"  "H3K9ME2"
[11] "H3K9ME3"   "HP1A"      "HP1B"      "MOD2"      "PC"
[16] "RNAPOL2"   "SU(HW)"    "SU(VAR)39"
> names(bg3.repset)
 [1] "ASH1"      "CP190"     "CTCF"      "EZ"        "H3K23AC"
 [6] "H3K27ME3"  "H3K36ME3"  "H3K4ME3"   "H3K79ME2"  "H3K9ME2"
[11] "H3K9ME3"   "HP1A"      "HP1B"      "MOD2"      "PC"
[16] "RNAPOL2"   "SU(HW)"    "SU(VAR)39"
> # Generate unified domain names
> color2domain <- c('Active','Active','HP1a','Polycomb','Boundaries')
> names(color2domain) <- c('lightgreen','purple','lightblue','yellow','grey')
> domains <- unique(s2names[s2names$Factor %in% mnames,c('Factor','Color')])
> domains$Domain <- color2domain[domains$Color]
> rownames(domains) <- domains$Factor
> # Compute distances
> mc.cores <- ifelse(.Platform$OS.type=='unix',8,1)
> d.s2 <- distGPS(GRangesList(s2.repset),metric='avgdist',mc.cores=mc.cores)
> d.bg3 <- distGPS(GRangesList(bg3.repset),metric='avgdist',mc.cores=mc.cores)
```

The core methodology of chroGPS, that is, computation of pairwise similarities (and thus, dis-similarities that can be interpreted as distances) between epigenomic factors based on their

binding profile overlaps, already provides some useful insight on relative configuration of epigenomic factor domains present in the data. In detail, this information can be compared across multiple datasets from which a rich number of common mapped factors is available, to assess correlation in vectors of similarities between pairs of analog factors or domains, in order to identify potentially strong biological or technical differences between them. We make use of this functionality using the *domainDist* function to assess the already observed general domain conservation between *Drosophila melanogaster* S2 and BG3 cell lines [**??**], and to see what happens when we artificially introduce a 'wrong' instance of factor EZ in the S2 dataset.

```
> # Compute inter-domain distances
> dd.s2 <- domainDist(as.matrix(d.s2),gps='factors',domain=domains$Color,type='inter',plot=FALSE)
> dd.bg3 <- domainDist(as.matrix(d.bg3),gps='factors',domain=domains$Color,type='inter',plot=FALSE)
> # Random seed
> set.seed(149)
> # Alterate s2
> s2.alt <- s2.repset
> df1 <- as.data.frame(s2.repset[['GAF']])[sample(1:(length(s2.repset[['GAF']])/2)),]
> df2 <- as.data.frame(s2.repset[['HP1B']])[sample(1:(length(s2.repset[['HP1B']])/2)),]
> s2.alt[['EZ']] <- GRanges(rbind(df1,df2))
> d.s2.alt <- distGPS(GRangesList(s2.alt),metric='avgdist',mc.cores=mc.cores)
> # Plot S2 vs BG3
> par(las=1,mar=c(4,8,4,4))
> mycors1 <- rev(diag(cor(as.matrix(d.s2),as.matrix(d.bg3))))
> barplot(mycors1,horiz=TRUE,xlim=c(0,1),main='S2 / BG3',col=domains[names(mycors1),'Color'],font=2)
> for (i in 1:length(summary(mycors1))) abline(v=summary(mycors1)[i],col=i,lwd=2,lty=3)
```

And now, lets see how this translates into a loss of correlation for both EZ distance vectors between both datasets.

```
> # Plot S2 Altered vs BG3
> par(las=1,mar=c(4,8,4,4))
> mycors2 <- rev(diag(cor(as.matrix(d.s2.alt),as.matrix(d.bg3))))
> barplot(mycors2,horiz=TRUE,xlim=c(0,1),main='S2 Altered / BG3',col=domains[names(mycors2),'Color'],font=2)
> for (i in 1:length(summary(mycors2))) abline(v=summary(mycors2)[i],col=i,lwd=2,lty=3)
```

6 Selecting candidate factors for designing de-novo epigenome mapping experiments.

Maximize chromatin domain identity: Chromatin domains offer an insightful and intuitive way to interpret epigenomic map conformation, by providing a biological context to factors based on functional relationships between them. When such information is available, a straightforward approach to select candidate factors for performing a de-novo epigenome mapping is to select those ones giving maximum robustness to their corresponding domain. This is easily achieved using the *rankFactorsbyDomain* function. We can use our chromatin color values as an alias to define chromatin domains. In this example we see how to perform a domain distance based selection for the HP1a repression considering a subset of 4 different factors.

```
> ## Rank Factors by Domain, using intra/inter domain distance
>
> data(s2)
> data(toydists)
> #d <- distGPS(s2,metric='avgdist',mc.cores=mc.cores) # Compute distances
> rownames(s2names) <- s2names$ExperimentName
> # Known domains
> # Call rankFactorsbyDomain for HP1a repression domain, select a combination of 4 factors
> library(caTools)
> rank.factors.4 <- rankFactorsbyDomain(d,s2names,ranktype='domainDist',selName='Color',selValue='lightblue',k=3
```

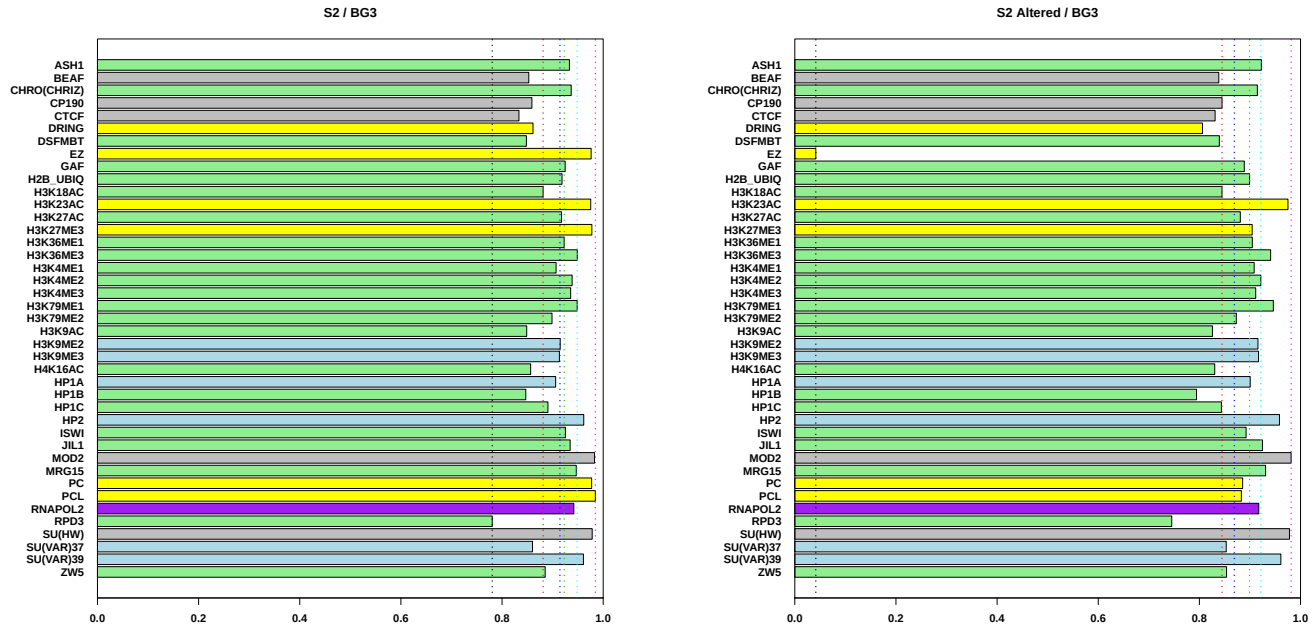

Figure 14: Correlation between intra-domain distance vectors from different datasets. Left: S2 vs BG3. Right: S2 vs BG3 with an artifially altered EZ replicate.

```
### Evaluating lightblue domain ###

> ddd <- as.data.frame(do.call(rbind,lapply(rank.factors.4,unlist)))
> ddd <- ddd[order(ddd$intra,decreasing=FALSE),]
> head(ddd)

                                                        inter
H3K9me3.S2 + HP1a_wa184.S2 + Su(var)3-9.S2           0.9405322
H3K9me2-Ab2-(new-lot).S2 + H3K9me3.S2 + Su(var)3-9.S2 0.9501828
H3K9me2-Ab2-(new-lot).S2 + HP1a_wa184.S2 + Su(var)3-9.S2 0.9461109
H3K9me2-Ab2-(new-lot).S2 + H3K9me3.S2 + HP1a_wa184.S2 0.9337194
                                                        intra
H3K9me3.S2 + HP1a_wa184.S2 + Su(var)3-9.S2           0.5442299
H3K9me2-Ab2-(new-lot).S2 + H3K9me3.S2 + Su(var)3-9.S2 0.5443542
H3K9me2-Ab2-(new-lot).S2 + HP1a_wa184.S2 + Su(var)3-9.S2 0.5476857
H3K9me2-Ab2-(new-lot).S2 + H3K9me3.S2 + HP1a_wa184.S2 0.5492476
```

Ranking factors based on functional relationship with genetic elements. Alternatively, or whenever domain information is not available, selection of candidate factors can be based upon conservation of functional relationships between epigenetic factors and genetic elements and how information on experimentally mapped factors has been successfully used to impute unknown ones [**?**]. The data for performing this analysis is that one used to generate chroGPS^{genes} maps (See Supplementary section 4), that is, a binary matrix of N genes (rows) per M columns (factors), where each cell equals 1 if that gene has an assigned binding site for that factor, and 0 otherwise. The *rankFactorsbyDomain* function provides methods based on linear and logistic regression to rank factors based on how accurately they can be predicted by others. At each iteration, the factor which can be best predicted by the rest is removed and prediction accuracies are recomputed. On the downside, these methods can be computationally intensive and we recommend using parallel computation or reducing the number of iterations if computing power is limited.

```
> # Using logistic regression to see how well some factors can be predicted by others
```

```
> glm.rank <- rankFactorsbyProfile(s2.tab,ranktype='glm',glm.threshold=0.75,mc.cores=mc.cores)

[1] "# Removing ASH1.Q4177.S2 factor with glm prediction score 1.000"
[1] "# Removing Chro.Chriz.BR.S2 factor with glm prediction score 1.000"
[1] "# Removing dRING.Q3200.S2 factor with glm prediction score 1.000"
[1] "# Removing EZ.Q3421.S2 factor with glm prediction score 1.000"
[1] "# Removing Ez.S2 factor with glm prediction score 1.000"
[1] "# Removing H2BK5ac.S2 factor with glm prediction score 1.000"
[1] "# Removing H3K4me2.Millipore.S2 factor with glm prediction score 1.000"
[1] "# Removing H3K4me2.ab.S2 factor with glm prediction score 1.000"
[1] "# Removing H3K4me3_S2.ChIP.chip factor with glm prediction score 1.000"
[1] "# Removing H3K4Me3.LP..S2 factor with glm prediction score 1.000"
[1] "# Removing H3K79Me2.S2 factor with glm prediction score 1.000"
[1] "# Removing H4K16ac.M..S2 factor with glm prediction score 1.000"
[1] "# Removing HP1a_552.S2 factor with glm prediction score 1.000"
[1] "# Removing HP1b..Henikoff..S2 factor with glm prediction score 1.000"
[1] "# Removing HP1a_wa184.S2 factor with glm prediction score 1.000"
[1] "# Removing HP2..Ab2.90..S2 factor with glm prediction score 1.000"
[1] "# Removing MBD.R2.Q4167..S2 factor with glm prediction score 1.000"
[1] "# Removing PCL.Q3412.S2 factor with glm prediction score 1.000"
[1] "# Removing Psc.S2 factor with glm prediction score 1.000"
[1] "# Removing mod2.2.VC.S2 factor with glm prediction score 0.998"
[1] "# Removing SU.HW..HB.S2 factor with glm prediction score 0.994"
[1] "# Removing Su.var.3.9.Q2598.S2 factor with glm prediction score 0.994"
[1] "# Removing Su.var.3.9.S2 factor with glm prediction score 0.994"
[1] "# Removing Pc.S2 factor with glm prediction score 0.979"
[1] "# Removing Chro.Chriz.WR.S2 factor with glm prediction score 0.961"
[1] "# Removing H3K9me2.Ab2..new.lot..S2 factor with glm prediction score 0.953"
[1] "# Removing JIL1_Q3433.S2 factor with glm prediction score 0.946"
[1] "# Removing Nurf301_Q4159.S2 factor with glm prediction score 0.951"
[1] "# Removing H3K36me1.S2 factor with glm prediction score 0.946"
[1] "# Removing CTCF.S2 factor with glm prediction score 0.946"
[1] "# Removing H4AcTetra.S2 factor with glm prediction score 0.948"
[1] "# Removing NURF301_Q2602.S2 factor with glm prediction score 0.942"
[1] "# Removing H3K9me3.S2 factor with glm prediction score 0.938"
[1] "# Removing RNA.pol.II..ALG..S2 factor with glm prediction score 0.933"
[1] "# Removing HP1a_wa191.S2 factor with glm prediction score 0.927"
[1] "# Removing WDS_Q2691.S2 factor with glm prediction score 0.931"
[1] "# Removing HP1c.Q4064.S2 factor with glm prediction score 0.916"
[1] "# Removing Su.var.3.7.Q3448.S2 factor with glm prediction score 0.916"
[1] "# Removing H3K9ac.S2 factor with glm prediction score 0.914"
[1] "# Removing SPT16_Q2583.S2 factor with glm prediction score 0.910"
[1] "# Removing H3K23ac.S2 factor with glm prediction score 0.912"
[1] "# Removing ZW5.S2 factor with glm prediction score 0.899"
[1] "# Removing H3K18ac.S2 factor with glm prediction score 0.897"
[1] "# Removing MBD.R2_Q2567.S2 factor with glm prediction score 0.897"
[1] "# Removing H2B.ubiq..NRO3..S2 factor with glm prediction score 0.888"
[1] "# Removing HP1b.Q4114.S2 factor with glm prediction score 0.884"
[1] "# Removing CTCF.VC.S2 factor with glm prediction score 0.873"
[1] "# Removing MRG15_Q2481.S2 factor with glm prediction score 0.867"
[1] "# Removing BEAF.HB.S2 factor with glm prediction score 0.863"
[1] "# Removing H3K9me2.antibody2.S2 factor with glm prediction score 0.858"
[1] "# Removing Pho.S2 factor with glm prediction score 0.858"
[1] "# Removing H3K27me3..Abcam2..S2 factor with glm prediction score 0.856"
[1] "# Removing Su.Hw..VC.S2 factor with glm prediction score 0.843"
[1] "# Removing HP1c..MO.462..S2 factor with glm prediction score 0.841"
[1] "# Removing RPD3.Q3451.S2 factor with glm prediction score 0.835"
[1] "# Removing H3K36me3.S2 factor with glm prediction score 0.835"
[1] "# Removing BEAF.70.S2 factor with glm prediction score 0.822"
[1] "# Removing H4K16ac.L..S2 factor with glm prediction score 0.820"
[1] "# Removing RNA.Pol.II..abcam..S2 factor with glm prediction score 0.813"
[1] "# Removing JHDM1_Q2634.S2 factor with glm prediction score 0.811"
[1] "# Removing H4K8ac.S2 factor with glm prediction score 0.805"
[1] "# Removing dMi.2_Q2626.S2 factor with glm prediction score 0.805"
```

```
[1] "# Removing ISWI_Q4095.S2 factor with glm prediction score 0.798"
[1] "# Removing CP190.VC.S2 factor with glm prediction score 0.785"
[1] "# Removing GAF.S2 factor with glm prediction score 0.766"
[1] "# Removing H3K9acS10P_.new_lot..S2 factor with glm prediction score 0.762"
[1] "# Removing dSFMBT.Q2642.S2 factor with glm prediction score 0.747"
[1] "# Removing CTCF.N_S2.ChIP.chip factor with glm prediction score 0.736"
[1] "# Removing JIL.1.Q4170..S2 factor with glm prediction score 0.727"
[1] "# Removing H3K27Ac.S2 factor with glm prediction score 0.732"
[1] "# Removing PR.Set7_Q3484.S2 factor with glm prediction score 0.695"
```

```
> # List of results indicating which factor is removed (best predicted by the rest) at each iteration
> names(glm.rank)
```

```
 [1] "ASH1.Q4177.S2"            "Chro.Chriz.BR.S2"
 [3] "dRING.Q3200.S2"           "EZ.Q3421.S2"
 [5] "Ez.S2"                    "H2BK5ac.S2"
 [7] "H3K4me2.Millipore.S2"     "H3K4me2.ab.S2"
 [9] "H3K4me3_S2.ChIP.chip"     "H3K4Me3.LP..S2"
[11] "H3K79Me2.S2"              "H4K16ac.M..S2"
[13] "HP1a_552.S2"              "HP1b..Henikoff..S2"
[15] "HP1a_wa184.S2"            "HP2..Ab2.90..S2"
[17] "MBD.R2.Q4167..S2"         "PCL.Q3412.S2"
[19] "Psc.S2"                   "mod2.2.VC.S2"
[21] "SU.HW..HB.S2"             "Su.var.3.9.Q2598.S2"
[23] "Su.var.3.9.S2"            "Pc.S2"
[25] "Chro.Chriz.WR.S2"         "H3K9me2.Ab2..new.lot..S2"
[27] "JIL1_Q3433.S2"            "Nurf301_Q4159.S2"
[29] "H3K36me1.S2"              "CTCF.S2"
[31] "H4AcTetra.S2"             "NURF301_Q2602.S2"
[33] "H3K9me3.S2"               "RNA.pol.II..ALG..S2"
[35] "HP1a_wa191.S2"            "WDS_Q2691.S2"
[37] "HP1c.Q4064.S2"            "Su.var.3.7.Q3448.S2"
[39] "H3K9ac.S2"                "SPT16_Q2583.S2"
[41] "H3K23ac.S2"               "ZW5.S2"
[43] "H3K18ac.S2"               "MBD.R2_Q2567.S2"
[45] "H2B.ubiq..NRO3..S2"       "HP1b.Q4114.S2"
[47] "CTCF.VC.S2"               "MRG15_Q2481.S2"
[49] "BEAF.HB.S2"               "H3K9me2.antibody2.S2"
[51] "Pho.S2"                   "H3K27me3..Abcam2..S2"
[53] "Su.Hw..VC.S2"             "HP1c..MO.462..S2"
[55] "RPD3.Q3451.S2"            "H3K36me3.S2"
[57] "BEAF.70.S2"               "H4K16ac.L..S2"
[59] "RNA.Pol.II..abcam..S2"    "JHDM1_Q2634.S2"
[61] "H4K8ac.S2"                "dMi.2_Q2626.S2"
[63] "ISWI_Q4095.S2"            "CP190.VC.S2"
[65] "GAF.S2"                   "H3K9acS10P_.new_lot..S2"
[67] "dSFMBT.Q2642.S2"          "CTCF.N_S2.ChIP.chip"
[69] "JIL.1.Q4170..S2"          "H3K27Ac.S2"
[71] "PR.Set7_Q3484.S2"
```

```
>
```

7 Comparing epigenomic factor maps.

Procrustes allows integration of epigenetic maps coming from different biological backgrounds, as well as adjustment of undesired biases due to technical effects [**??**]. Furthermore, it can be used to identify differences between epigenomic factor maps generated at different conditions (healthy/disease, control/treated), coming from distinct biological backgrounds, or being snapshots of different points in time, such as developmental stages or timings in origins of replication. We will focus on this elements to illustrate comparison of chroGPS^{factors} maps.

Origins of replication are particular genomic sequences at which replication of DNA starts in living eucharyote and prokaryote organisms, and of DNA-RNA in viruses. These sequences are recognized by specific proteins, that recognize, unwind and begin to copy the genomic sequence. In the following steps we illustrate how to use chroGPS to compare the epigenomic landscape at these different time points.

First, we start by loading Origins of Replication data from *Drosophila melanogaster* S2 cells available in modENCODE [**?**], and contains location for Origins of Replication at Early, Early Mid, Late Mid and Late cell replication time points, stored in four different BED files. The other data to perform our study will be our already familiar collection of epigenomic elements (genomic intervals) in all conditions we would like to compare. In our case we will use *Drosophila melanogaster* S2 modENCODE binding sites and will filter them according to the different collections of Origins of Replication regions we just loaded. This can be easily performed using basic IRanges overlap operations.

```
> # Intersect s2 with repliSeq, filter peaks by overlap with origin set
> # Assuming the 'orig' objects is a RangedDataList with modEncode S2 origins of Replication for Early to Late t
> # Assuming S2 has the 'full' S2 dataset used in Font-Burgada et al. 2014
> #s2.origs <- lapply(orig,function(o) GRangesList(mclapply(as.list(s2),function(x) x[x %over% o,],mc.cores=mc.c
>
> # Make distance sets
> #d.origs <- lapply(s2.origs,function(x) distGPS(x,metric='avgdist',mc.cores=mc.cores))
> #m.origs <- lapply(d.origs,mds,type='isoMDS')
>
> # Now load precomputed data
> data(s2)
> data(repliSeq)
> library(gplots)
> # Modify colors and add some transparency
> fnames <- s2names$Factor
> s2names$Color[s2names$Color=='grey'] <- 'orange'
> fcolors <- paste(col2hex(s2names$Color),'BB',sep='')
> bcolors <- paste(col2hex(s2names$Color),'FF',sep='')
```

The next step is to generate regular chroGPS^{factors} maps of each joint dataset. Map comparison is performed using Procrustes to measure and rank changes between two given maps in an unbiased manner, which is done automatically with the *diffFactors* function. This function performs the dual task of finding an optimal adjustment in shift, scale and rotation so that both maps are matched as much as possible while maintaining relative distances between the respective factors of each one and providing the user with the internal sum of error metrics of the Procrustes algorithm. As a result, we obtain both a graphical 2D chroGPS^{factors} map where distances between replicates of the same element in both backgrounds are highlighted, and a ranked list of Procrustes errors for all common factors involved in the map. Statistical significance of the observed changes can be assessed via overlap permutation tests. Figure **??** illustrates results for Procrustes differential analysis of the transition between Early-Mid and Late time points.

```
> # Select time points to compare
> m1 <- m.origs[['Early.Mid']]
> m2 <- m.origs[['Late']]
> ## Perform differential Procrustes analysis
> df <- diffFactors(m1,m2)
> ## Plot both maps before and after adjustment
> m3 <- df$mds3
> par(mfrow=c(1,2))
> plot(0,xlim=c(-1,1),ylim=c(-1,1),xlab='',ylab='',xaxt='n',yaxt='n',col='NA')
> segments(m1@points[,1],m1@points[,2],m3@points[,1],m3@points[,2],col='red')
> par(new=TRUE)
```

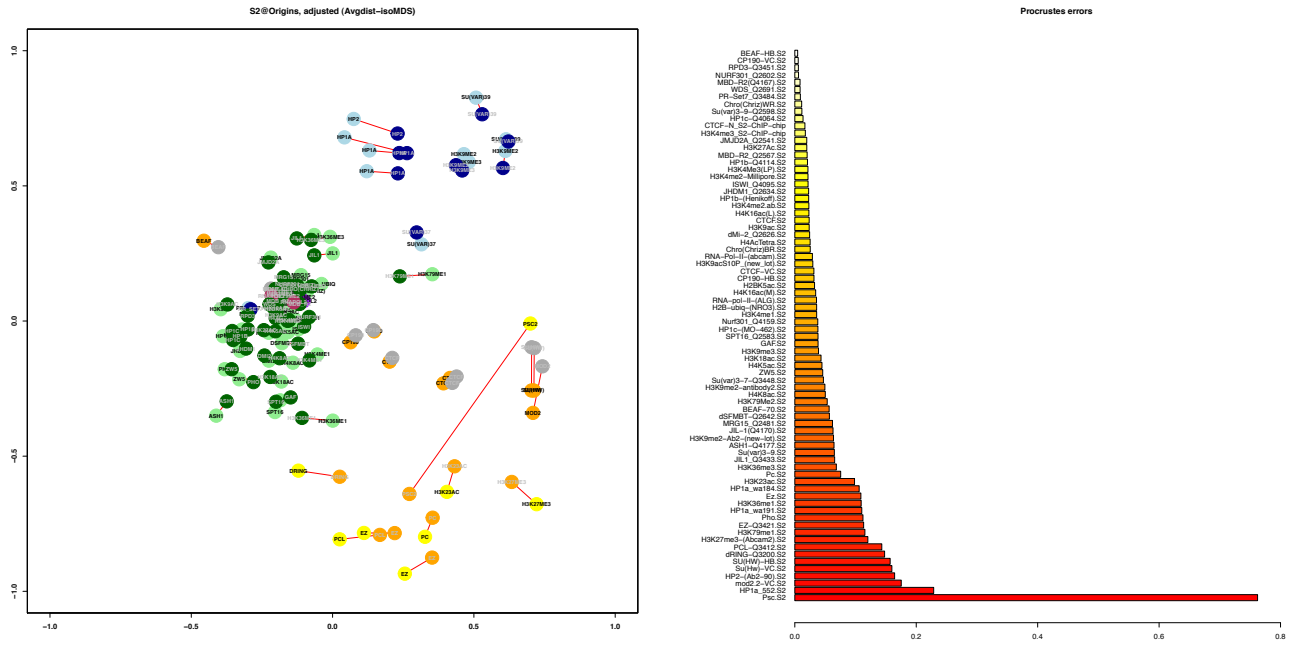

Figure 15: Factor differential map and Procrustes errors for pair of epigenetic factor replicates between the compared Origins of Replication time points.

```
> plot(m1,drawlabels=TRUE,labels=s2names$Factor,point.pch=19,point.cex=4,text.cex=0.75,point.col=s2names$Color,m
> par(new=TRUE)
> plot(m3,drawlabels=TRUE,labels=s2names$Factor,point.pch=19,point.cex=4,text.cex=0.75,point.col=s2names$Darkcol

> ## Plot summary of errors by point
> par(las=1,mar=c(4,12,4,4));
> pp <- df$procrustes
> barplot(sort(residuals(pp),decreasing=TRUE),horiz=TRUE,xlim=c(0,max(residuals(pp))+.1),col=heat.colors(length(
```

8 Comparing epigenomic gene maps.

Generation of differential chroGPS^{genes} maps to compare two epigenomic data sets is performed in a very similar way to the ones presented above. First, common factors between both backgrounds to compare are selected, usually after performing some operation to unify factor replicates when present. Then, for genes with at least one epigenetic mark in each background we compute pairwise distances between their epigenetic profiles using our metric of choice and a 2 dimensional map is generated via MDS. This allows to keep trace both of the epigenetic and background identity for each gene in the analysis. Since genes can be identified easily on the map by means of its epigenetic profile, it is straightforward to know which genes from different conditions present the same or very similar profiles (thus they are located in the same exact point or very close in the map), and which ones present significant differences (and therefore differ strongly in their location).

Downstream functional analysis of this differential map is essentially done in the same way as in a regular chroGPS^{genes} one. Briefly, hierarchical clustering is performed over the mathematical distances computed between each pair of unique epigenetic profiles. The identified clusters in this first step are then subject to an unsupervised merging process in order to further refine cluster definition and reduce granularity inherent to this method. Finally, after a final clustering configuration is obtained, each gene is assigned a posterior probability of correct classification

by means of Bayesian density estimation procedures. In this way, we present not just a (simple and elegant) method to obtain an average robustness / reproducibility score for each one of the clusters and also for the global clustering solution, but also to obtain a posterior probability value indicating how sure we are that a given gene is well located within its assigned cluster. This strategy proves very useful also not just to refine selection of those genes changing clearly from any two given clusters (i.e. genes suffering a strong change in epigenomic identity), but can also be used with differential maps to assign a probability score in order to rank gene changes involved in cluster changes of interest, providing us with an intuitive method to select potential candidate genes for further analysis in a scenario where two different backgrounds want to be compared.

```
> # Summarize factor replicates with method 'any' so that 1 replicate having the mark is enough
> # Assuming s2.tab and bg3.tab contain the full datasets for dm3 genome and all factors used in
> # Font-Burgada et al.# See https://github.com/singlecoated/chroGPS2 for available full datasets
> # to download
>
> # Not run
> # s2.tab <- mergeReplicates(s2.tab,s2names$Factor,'any')
> # bg3.tab <- mergeReplicates(bg3.tab,bg3names$Factor,'any')
>
> # Join, use common factors. Then use common genes only from those that have at least one mark in both s2 and b
> # x <- combineGenesMatrix(s2.tab,bg3.tab,'S2','BG3')
>
> # Build map and cluster as always
> # d <- distGPS(x,metric='tanimoto',uniqueRows=TRUE)
>
> # m <- mds(d,type='classic',splitMDS=TRUE,split=0.16,mc.cores=mc.cores)
> # mm <- mds(d,m,type='boostMDS',samplesize=0.005,mc.cores=mc.cores)
>
> # Cluster
> # h <- hclust(as.dist(d@d),method='average')
> # clus <- clusGPS(d,mm,h,k=max(cutree(h,h=0.5)),ngrid=10000,mc.cores=mc.cores,recalcDist=FALSE,verbose=FALSE)
> # clus.merged <- mergeClusters(clus,brake=0,mc.cores=mc.cores)
> # clus
> # clus.merged
>
> # Epigenetic cluster profiles
> # pc <- profileClusters(x,clus.merged,normalize=TRUE)
> # pheatmap(pc,trace='none',scale='none',col=bluered(100))
```

The usual scenario in which to perform a differential chroGPS^{genes} map is that one involving studying epigenetic profiles over genes in two different conditions, biological or technical backgrounds from the same species, even though one could use the same methodology to address for instance changes in regulatory mechanisms in promoter vs coding regions or origins of replication, by simply recomputing binding site assignments under different genomic locations, etc. The basic elements to generate this map are the two collection of epigenetic elements which we want to compare at our desired level to use for generation of the necessary binary matrices described previously, or in its defect, two already computed matrices of 0s and 1s representing epigenetic profiles for each genetic element in each condition. Only common epigenetic factors mapped in both conditions will be taken into account (and therefore, only common genetic elements with at least 1 mapped factor in each condition are used). The *combineGenesMatrix* function takes raw *genes* x *factors* matrices for both backgrounds and returns a unique matrix containing unique epigenetic profiles for them.

```
> # Perform differential analysis
> # x.diff <- diffGenes(x,mm,clus.merged,label.x='S2',label.y='BG3',clusName=clusNames(clus.merged)[1],fdr=TRUE,
>
```

```
> # Select genes changing clusters with FDR 0.05
> # xx.diff <- x.diff[x.diff$ClusID.S2!=x.diff$ClusID.BG3 & x.diff$FDR.S2<0.25 & x.diff$FDR.BG3<0.25,]
> # xx.diff$CC <- paste(xx.diff$ClusID.S2,xx.diff$ClusID.BG3,sep='.')
> # head(sort(table(xx.diff$CC),decreasing=TRUE))
>
>
> # Perform enrichment test using getEnrichedGO from chippeakanno package for cluster transitions 2.9 and 5.2
> # library(ChIPpeakAnno)
> # library(org.Dm.eg.db)
>
> # enriched.GO <- lapply(c('2.9','5.2'),function(cc) {
> #        fbid <- as.character(xx.diff$geneid[xx.diff$CC==cc])
> #            if (length(fbid)>=25)
> #                        ans <- getEnrichedGO(annotatedPeak=fbid,orgAnn='org.Dm.eg.db',maxP=0.05,multiAdjMetho
> #            else ans <- NULL
> #            return(ans)
> #    })
>
> # names(enriched.GO) <- c('2.9','5.2')
> # enriched.GO <- enriched.GO[unlist(lapply(enriched.GO,length))>0]
> # enriched.GO <- lapply(enriched.GO,function(x) lapply(x,function(y) unique(y[,-ncol(y)])))
> # lapply(enriched.GO,head)
>
> # Plot results with diffGPS.plot function
> # res.sel <- res[res$ClusID.S2!=res$ClusID.BG3,]
>
> # Plot
> # diffGenes.plot(x,mm,clus.merged,res.sel,transitions='10.2',label.x='S2',label.y='BG3',fdr1=0.25,fdr2=0.25)
```

Once the combined matrix is obtained, a regular chroGPS^{genes} map is produced as described previously, and epigenetic cluster characterization can be performed with the clusGPS, mergeClusters and profileClusters function as with single chroGPS^{genes} maps. Afterwards, the diffGenes function will identify genes presenting statistically significant epigenetic profile transitions between both clusters, and results will be returned as a data frame with all the necessary information to perform further downstream analysis, such as Gene Ontology enrichment. Results can be presented using the available diffGenes.plot function, and individual map points and differential analysis results are easily to obtain from the returned objects and can be used in custom and interactive graphical outputs.

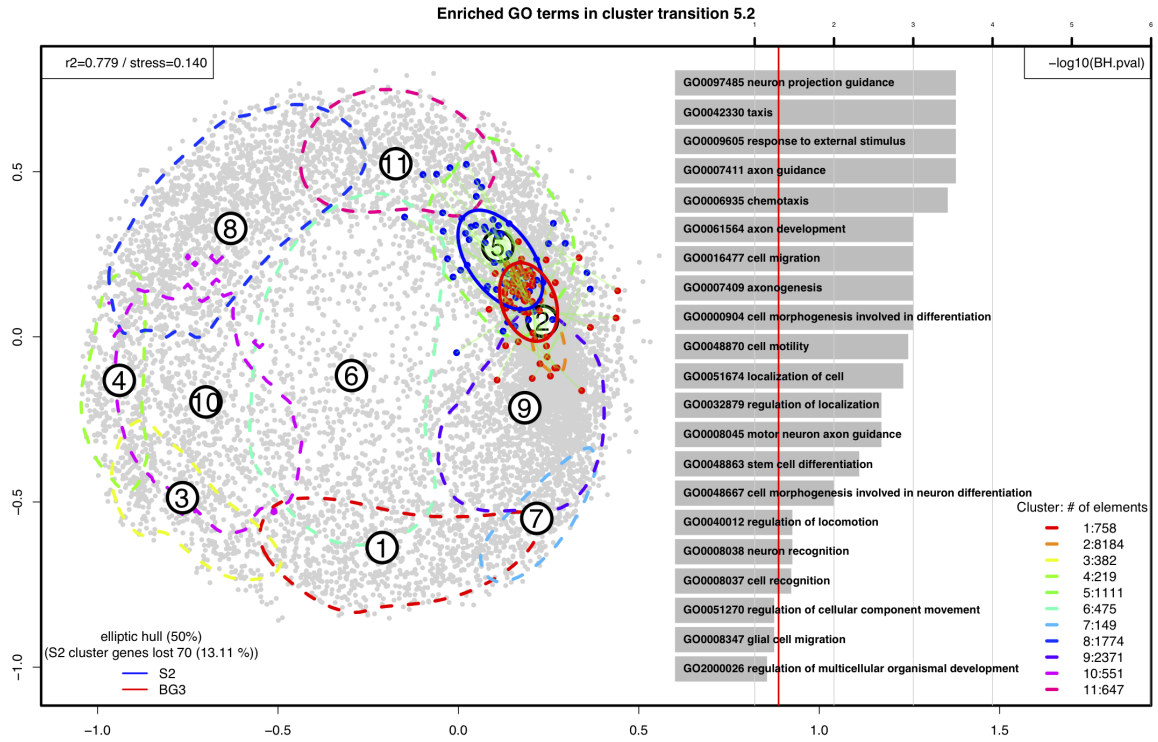

Figure 16: Differential chroGPS^{genes} map for the Drosophila melanogaster S2 and BG3 cell lines with highlighted cluster transitions of interest.

9 Beyond R: generating interactive chroGPS maps for Cytoscape, Plotly, ...

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`, [Shannon et al., 2003]. 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.
Additionally, all the objects containing the generated maps in 2 and 3 dimensions can be used to obtain the individual graphical coordinates for all points in the map using the provided function *getPoints*, in order to produce custom graphics, or to use the obtained maps in interactive HTML outputs using Plotly.

```
> # For instance if mds1 contains a valid chroGPS$^{factors}$ map.
> # gps2xgmml(mds1, fname='chroGPS_factors.xgmml', fontSize=4,
> # col=s2names$Color, cex=8)
> # And use Cytoscape -> File -> Import -> Network (Multiple File Types)
> # to load the generated .xgmml file
```

Or for instance, if we would like to generate an interactive, light and shareable view of our 2D or 3D chroGPS map using Plotly, we can use the following code as an example, to replicate a map like the one shown in Figure **??**, something which greatly enhances sharing visually atractive and interactive chroGPS maps, and also allow integration of further annotation information in HTML format.

```
> # Not run
> # Requires Pandoc
> # library(plotly)
> # library(gplots)
> # For instance if mds1.3d contains a valid 3 dimensional chroGPS$^{factors}$ map from our S2 example data.
> # color2domain <- c('Active','RNAPol2','HP1a','Polycomb','Boundaries')
> # names(color2domain) <- c('lightgreen','purple','lightblue','yellow','grey')
> # xx <- as.data.frame(getPoints(mds1.3d))
> # colnames(xx) <- c('x','y','z')
> # xx$color <- s2names$Color
> # xx$domain <- as.factor(color2domain[xx$color])
> # ids <- as.character(s2names$Factor)
> # p <- plot <- ly(xx,x=~x,y=~y,z=~z,color=~domain,colors=col2hex(unique(xx$color)),text=ids) %>%
> #     add <- markers() %>%
> #     layout(title='S2 3D plotLy example',
> #              scene = list(xaxis = list(title = 'X'),
> #                    yaxis = list(title = 'Y'),
> #                    zaxis = list(title = 'Z')))
> #htmlwidgets::saveWidget(as <- widget(p), "../reports/chrogps_3d_plotly.html") # export for offline use, requi
```

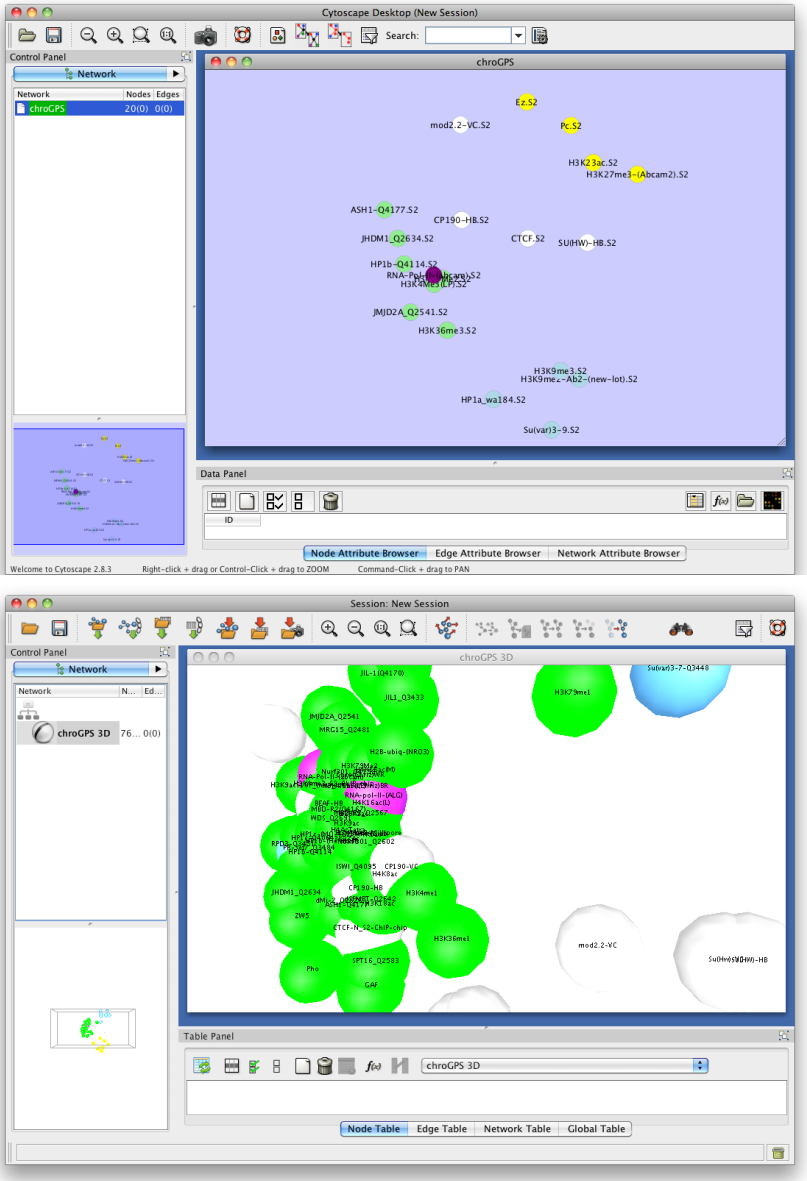

Figure 17: chroGPS^{factors} network exported and visualized in Cytoscape. Top: 2D. Bottom: 3D.

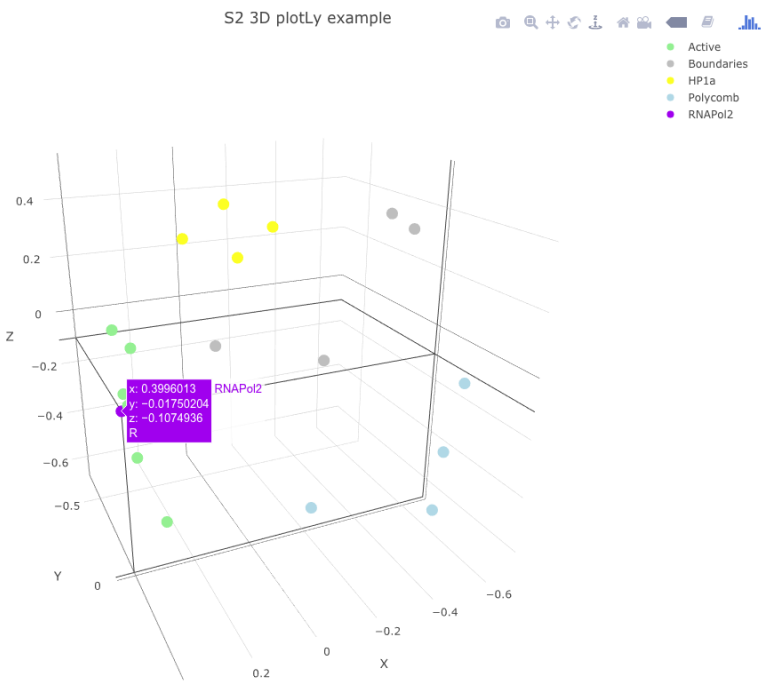

Figure 18: chroGPS^{factors} map exported and visualized as an HTML5 3d graphic in PlotLy.

References

J. Font-Burgada, O. Reina, D. Rossell, and F. Azorin. chrogps, a global chromatin positioning system for the functional analysis and visualization of the epigenome. *Submitted*, 2013.

P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.*, 13(11):2498–2504, Nov 2003.

W. N. Venables and B. D. Ripley. *Modern applied statistics with S*. Springer, 4th edition, August 2002. ISBN 0387954570.

L.J. Zhu, C. Gazin, N.D. Lawson, H. Pagès, S.M. Lin, D.S. Lapointe, and M.R. Green. ChIPpeakAnno: a bioconductor package to annotate chIP-seq and chIP-chip data. *BMC Bioinformatics*, 11:237, 2010.