Annotation-Driven Class Discovery

User's Guide to the Bioconductor package *adSplit*

Claudio Lottaz[1], Joern Toedling and Rainer Spang

Max Planck Institute for Molecular Genetics
Computational Diagnostics, Department for Computational Molecular Biology
Ihnestr. 63-73, D-14195 Berlin, Germany

Technical Report
Nr. 2005/02

Abstract

This is the vignette of the Bioconductor compliant package *adSplit*. We describe our implementation of *annotation- driven clustering* for microarray gene expression profiling studies.

---

[1]Corresponding author: `claudio.lottaz@molgen.mpg.de`

Contents

Chapter 1

Introduction

Background: Clustering algorithms are widely used in the analysis of microarrays. In clinical studies, they are not only used to cluster genes into groups of co-regulated genes, but also for clustering patients, and thereby defining novel disease entities based on gene expression profiles. Several distance based cluster algorithms have been suggested, but little attention has been paid to the choice of the metric on patient space. Even when using the Euclidean metric, including and excluding genes from the analysis leads to different distances between the same objects, and consequently to different clustering results.

Methodology: In this package, we implement a novel algorithm for investigating the dependency of clustering results on the choice of metric supporting genes. Our method combines expression data and functional annotation data. According to gene annotations, a list of candidate gene sets with a unique functional characterization is generated. Each set defines a metric on patient space, and consequently a clustering of patients. Based on a novel significance measure for clusterings, this list is filtered. Significant clusterings are reported together with the underlying gene sets and their functional definition.

Intended results: Our method reports clusterings defined by biologically focussed sets of genes. In our annotation driven clusterings, we have observed rediscoveries of clinically relevant patient subgroups through biologically plausible sets of genes. Hence, we conjecture that our method has the potential to reveal clinically relevant classes of patients in an unsupervised manner.

The algorithmic concept: We suggest a systematic approach to gene selection in an unsupervised setting. We describe an algorithm that produces a list of alternative clusterings using a variety of different metrics on patient space. While all metrics are of the Euclidean type, they differ in the set of genes used for characterizing the patient profiles. We derive candidate gene sets from functional annotation data, and filter the list by a novel significance measure for clustering strength. For practical use, it is desirable to have functional rationales characterizing clusterings. For instance, clusterings related to proliferation or apoptosis. To this end, we define candidate gene sets using functional

annotations from the Gene Ontology [1] and from the Kyoto Encyclopedia of Genes and Genomes [7].

The algorithm in a nutshell:

Annotation-driven splits specific to the corresponding gene set are computed using the k-means algorithm [5]. We use functional annotations from chip-specific meta-data packages provided in Bioconductor [3]. The quality of any clusterings is assessed using the *diagonal linear discriminant* (DLD) score [8]. In order to determine the statistical significance of a score, we also compute DLD scores for restricted metrics resulting from randomly chosen gene sets. Empirical p-values are calculated and false discovery rates (FDR) computed according to Benjamini Hochber [2]. Finally, we filter the list of clusterings for minimal subgroup size and to control the FDR. In a nutshell, the algorithm consists of the following steps:

For each biological term / pathway of interest, denoted B_i:

1. Find all n_{B_i} genes annotated to B_i and discard all others.

2. Perform 2-means clustering of reduced expression matrix. This yields an annotation-driven clustering C_{B_i}.

3. Compute DLD score $S(C_{B_i})$ for this clustering.

4. Draw 10000 random gene sets of size n_{B_i} from the set of all measured genes. For each of these random gene sets, compute steps 2 and 3. This yields a vector $\mathbf{r}_{n_{B_i}}$ of 10000 scores.

5. Assign an empirical p-value to the original clustering, denoting the proportion of entries of $\mathbf{r}_{n_{B_i}}$ being greater or equal than $S(C_{B_i})$.

6. Correct the empirical p-values for multiple testing according to Benjamini-Hochberg [2]. The resulting q-values are used to control the false discovery rate of a *splitSet*.

The structure of the vignette: The described package contains functions to facilitate these steps. The first chapter describes, how to compute a split for a given term of intrest. the second shows how to compute a random distribution of DLD-scores to judge the significance of an annotation-driven split. Finally, the last chapter describes how to collect results for many terms of interest and illustrate the achieved results.

Chapter 2

Annotation-Driven Splits

Before starting any analysis you have to load the *adSplit* package in your R session as follows:

```
> library(adSplit)
```

For illustration of *adSplit* usage we use the Golub data set on acute leukemia [4] as it is stored in the *golubEsets* experimental data package. For preparing the data, issue the following commands:

```
> library(golubEsets)
> data(Golub_Merge)
```

2.1 Initialize k-means with divisive hierarchical cluster centroids

K-means clustering critically depends on its initialization step. We derive an initialization based on the first split of a divisive hierarchical clustering (Chapter 6 in [9]). Of the resulting two clusters, we compute centroids which provide the starting points for the k-means algorithm [10]. This has been shown to outperform standard k-means with random starting points [11]. In fact, k-means is used to refine individual clusters and to correct inappropriate assignments made by the hierarchical method.

We have packed this procedure into the function `diana2means` of the here described package. In addition this function computes DLD-scores for the generated splits using the implementation taken from the ISIS package [8]. Actually, the determined split is only returned when return.cut is set to true. The following snipit of code uses the 10% most variable genes in the Golub-dataset to generate a split.

```
> e <- exprs(Golub_Merge)
> vars <- apply(e, 1, var)
```

```
> e <- e[vars > quantile(vars,0.9),]
> diana2means(e)

[1] 9.983408

> diana2means(e, return.cut=TRUE)

2-means split holding  72  elements
   distribution is:  52 20
   DLD-score:  9.983408
```

This function returns a single number representing the splits DLD-score as default, when the argument `return.cut` is set to `FALSE`. Otherwise an object of class *split* holding the list elements `cut` and `score` is returned. For instance, the the split attributions can be extracted as follows:

```
> x <- diana2means(e, return.cut=TRUE)
> x$cut

     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]   1    0    0    1    0    0    0    1    1    0     1     1
     [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22]
[1,]    0     0     0     0     0     1     1     1     0     0
     [,23] [,24] [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32]
[1,]    1     0     1     1     1     1     1     0     1     0
     [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42]
[1,]    1     1     0     0     0     0     0     0     1     0
     [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50] [,51] [,52]
[1,]    0     0     0     0     0     0     0     0     0     0
     [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62]
[1,]    0     0     0     0     0     0     0     0     0     0
     [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72]
[1,]    1     0     0     0     0     0     0     0     0     0
```

For the computation of the DLD-score, this function takes into account the best scoring `ngenes` genes (rows). In order to avoid excessive influence of single strikingly well separating genes, the `ignore.genes` argument can be used to dismiss the strongest genes from the score.

2.2   Annotation-driven splits

The central function for the generation of annotation driven splits is called `adSplit`. It calls the above described function `diana2means` on a restricted set of genes. This function has a series of arguments, but its basic call is as follows:

```
> adSplit(Golub_Merge, "GO:0006915", "hu6800")

Evaluating identifier GO:0006915 with 1636 probesets...
 -> skipped, too many probes associated (1636)
Empty split set
```

This command generates an annotation-driven split for the term "apoptosis" encoded by the identifier "GO:0006915". The string given as the chip's name is used to load the annotation meta-data. Thus *adSplit* expects a library of the same name to be installed, where it looks for the hashes `<chip-name>GO2ALLPROBES` and `<chip-name>PATH2PROBE` as they are provided by Bioconductor meta data packages. If we issue a similar command for the term "signal transduction" (GO:0007165), the following happens:

```
> adSplit(Golub_Merge, "GO:0007165", "hu6800")

Evaluating identifier GO:0007165 with 4726 probesets...
 -> skipped, too many probes associated (4726)
Empty split set
```

This term has too many associated genes. Hence, it is skipped. In order to generate splits related to more generic terms, we can provide an explicit maximum limit for the amount of annotated genes to terms of interest. For instance:

```
> adSplit(Golub_Merge, "GO:0007165", "hu6800", max.probes=7000)

Evaluating identifier GO:0007165 with 4726 probesets...
Annotation-driven split set
  holds 1 split on 72 elements
  associated to: signal transduction (GO:0007165)
  object distribution is: 52 20
  score is: 13.49333
  no empirical p-values computed
```

This command generates the wanted split based on more than 2000 genes. `adSplit` returns an object of class *splitSet* with the following list elements:

1. `cuts`: a matrix of split attributions. One row per annotation identifier (GO term or KEGG pathway for which a split has been generated. One column per object in the dataset.

2. `score`: one score per generated split

3. `pvalue`: one empirical p-value per generated split, or `NULL`.

The object may also be empty in which case all elements are `NULL`.

Chapter 3

Empirical p-Values for Splits

In order to generate empirical p-values for a given annotation-driven gene set, we suggest to sample random gene sets of the same size and apply the split generation algorithm implemented in `diana2means` to all of these gene sets. DLD-scores computed for the random gene sets are used to approximate the score's null distribution. The fraction of the scores from this null distribution, which are higher than the observed score for the gene set with common annotation, is used as empirical p-value. Some details on how to compute these p-values with *adSplit* are collected in this chapter.

3.1 Drawing random gene sets

The first step needed for the random sampling is drawing a given number of probe-sets measured in dataset. In order to reflect one obvious characteristic of annotation based gene sets, once we have drawn one probe set at random, we always include all other probe-sets representing the same gene into the random selection. In order to speed up the repeated drawing procedure, we use a hash containing one entry per EntrezGene identifier holding all associated probe-sets. This environment is generated from the meta-data package ENTREZID-hash as follows:

```
> EID2PSenv <- makeEID2PROBESenv(hu6800ENTREZID)
```

The returned hash is used as follows to draw random sets of probe-sets:

```
> drawRandomPS(10, EID2PSenv, ls(EID2PSenv))

             2691                696               3678               8111
  "L00137_cds1_at"        "U39576_at"        "X06256_at"        "U48405_at"
            90411              10578               4869               8243
      "M23161_at"        "M85276_at"        "M23613_at"      "S78271_s_at"
```

```
          5087               4077
     "M86546_at"        "D30756_at"
```

drawRandomPS returns a named vector holding random probe-set identifiers. The names of this vector corresponds to the associated LOCUSLINK identifiers.

3.2 Generating DLD-score distributions

The functions for drawing random sets of probe-sets described in the previous section are combined with diana2means to generat null-distributions of DLD-scores. This is implemented in the function randomDiana2means which is used as follows:

```
> scores <- randomDiana2means(20, exprs(Golub_Merge), "hu6800", ndraws = 1000)

  determining 1000 random DLD-scores with 20 probe sets each (wait for 10 dots)
  ..........
```

The form of this distribution is skewed. The parameter ignore.genes changes this shape towards a more symmetric shape as shown in the next figure. This is intuitive, since we remove genes which drive DLD-scores and thus approach more closely a random distribution.
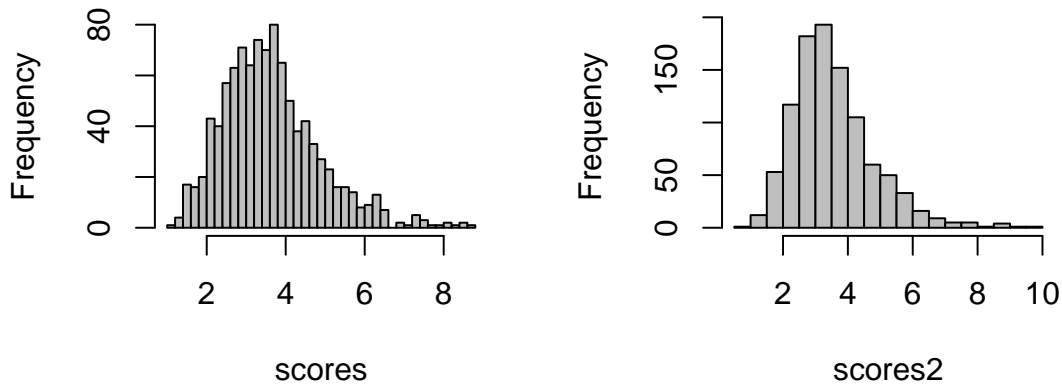
```
> scores2 <- randomDiana2means(20, exprs(Golub_Merge), "hu6800",
+                               ndraws = 1000, ignore.genes=5)

  determining 1000 random DLD-scores with 20 probe sets each (wait for 10 dots)
  ..........
```

```
> par(mfrow=c(1,2))
> hist(scores,  nclass=30, main="", col="grey")
> hist(scores2, nclass=30, main="", col="grey")
```

The left histogram here is the score distribution when including all genes, the right one results from exclusion of best scoring genes.

3.3 Adding empirical p-values while generating annotation driven splits

Finally, we can use the distribution generated with random sets of probe sets to compute empirical p-values. This is done by the `adSplit` function when `B` is specified, the number of samplings to be used.

```
> glutamSplits <- adSplit(Golub_Merge, "KEGG:00251", "hu6800", B=1000)

Evaluating identifier KEGG:00251 with 0 probesets...
 -> skipped, too few probes associated (0)
```

This call returns an object of class *splitSet* with an additional entry called `pvalue`. The print method for the *splitSet* gives a summary on sets of splits and some additional information, if a split set contains only 1 split:

```
> print(glutamSplits)

Empty split set
NULL
```

Chapter 4

Working with Split-Sets

4.1 Generating split-sets

The function `adSplit` accepst more than one annotation identifier in one call. In this case it generates splits for each of the provided identifier and collects the results into one single returnde object. For instance:

```
> x <- adSplit(Golub_Merge, c("GO:0007165","GO:0006915"), "hu6800", max.probes=7000)

Evaluating identifier GO:0007165 with 4726 probesets...
Evaluating identifier GO:0006915 with 1636 probesets...

> print(x)

Annotation-driven split set
  holds 2 splits on 72 elements
  scores range is: 12.3339 13.49333
  no empirical p-values computed
```

If the user doesn't want to specify few terms of interst in this fashion, he/she may also provide one of the following specially treated identifiers:

1. `GO`: all available GO terms are used as gene set candidates.

2. `KEGG`: all available KEGG pathways are used as gene set candidates.

3. `all`: both these sets of annotation identifiers are used.

For instance, the following command determines all splits driven by KEGG pathways without sampleing random gene lists for significance analysis:

```
> x <- adSplit(Golub_Merge, "KEGG", "hu6800")
```

```
> print(x)

Annotation-driven split set
  holds 131 splits on 72 elements
  scores range is: 3.225585 13.3296
  no empirical p-values computed
```

However, if empirical p-values are computed by sampling random gene lists, multiple testing is an issue to be considered. We use the *multtest* to correct our p-values and thus computing false discovery rates by the method suggested by Benjamini-Hochberg. The corresponding q-values are stored in the result's list element called `qvalues`. You may consider for illustration the precomputed object `golubKEGGSplits`:

```
> data(golubKEGGSplits)
> print(golubKEGGSplits)

Annotation-driven split set
  holds 70 splits on 72 elements
  scores range is: 3.382672 17.31385
  empirical p-values range is: 0.005 0.955
  q-value range is: 0.1633333 0.955

> summary(golubKEGGSplits$qvalues)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1633  0.4375  0.6681  0.6098  0.7745  0.9550
```

This object has been precomputed with the following command:

```
> golubKEGGSplits <- adSplit(Golub_Merge, "KEGG", "hu6800", B=1000)
```

It thus contains all splits driven by KEGG pathways and holds empirical p-values deduced from 1000 random gene lists. Pathways which have fewer than 20 genes associated are thereby skipped from the analysis.

4.2 Graphical illustrations

For showing the illustration utilities implemented in *adSplit*, we use the precomputed object `golubKEGGSplits` included in the *adSplit* package.

The `golubKEGGSplits` object contains 70 splits and corresponding empirical p-values and corrected q-values. The split set is ordered according to p-values such that the first entries are the most significant ones. In order to get an overview of whether significant splits and how many of them are generated, the package *adSplit* offers a histogram method for objects of class *splitSet* called as follows:

```
> data(golubKEGGSplits)
> hist(golubKEGGSplits)
```

Distribution of p–Values
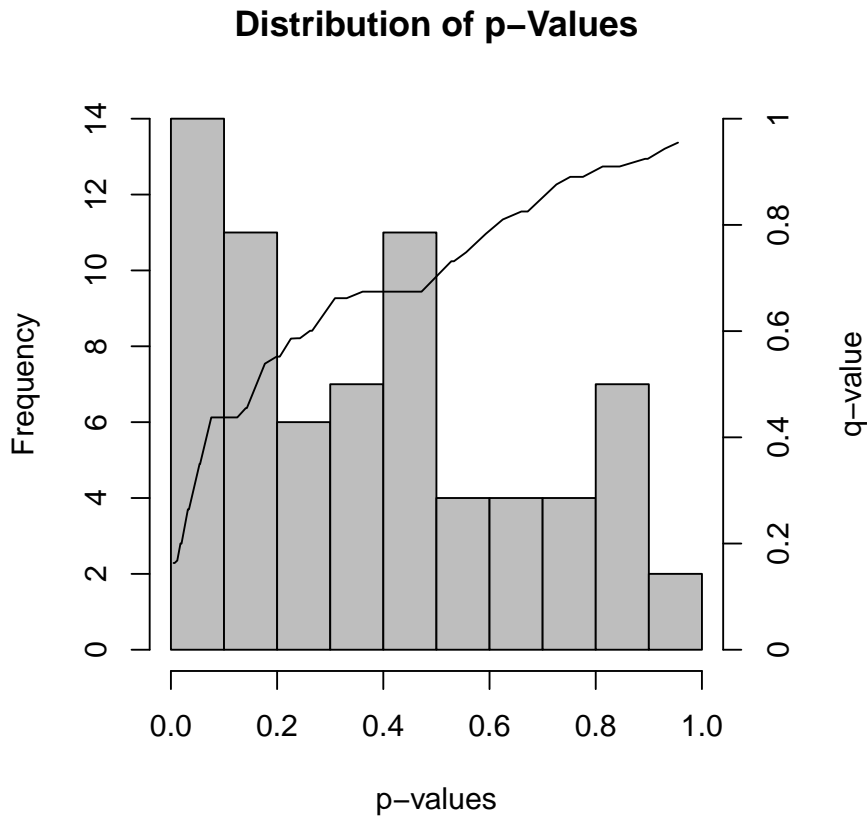
In this histogram the empirical p-values are drawn as usual in a histogram. In addition, the corresponding q-values corrected for multiple testing are plotted as a line into the histogram. The corresponding scale is shown to the left of the plot.

An `image` method on the same objects can be used to get an actual representation of all splits generated. A filter argument called `filter.fdr` is used to focus on splits with a low false discovery rate. The following call requires a set of splits with less then 30% expected false positives.

```
image(golubKEGGSplits, filter.fdr=0.3)
```
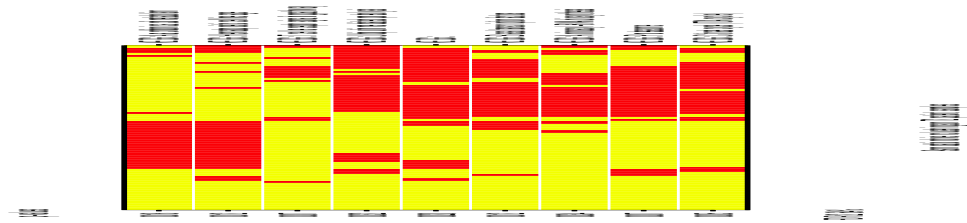
In this image, each column corresponds to a patient and each row corresponds to an annotation. The colors represent to which group the corresponding patient is attributed with respect to the corresponding annotation. This image is clustered in both directions in order to bring similar splits as well as similar patients close together.

Acknowledgements

Chapter 5

Bibliography

[1] M Ashburner, CA Ball, JA Blake, D Botstein, H Butler, JM Cherry, AP Davis, K Dolinski, SS Dwight, JT Eppig, MA Harris, DP Hill, L Issel-Tarver, A Kasarskis, S Lewis, JC Matese, JE Richardson, M Ringwald, GM Rubin, and G Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, 25(1):25–29, May 2000.

[2] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, 57(1):289–300, 1995.

[3] RC Gentleman, VJ Carey, DM Bates, B Bolstad, M Dettling, S Dudoit, B Ellis, L Gautier, Y Ge, J Gentry, K Hornik, T Hothorn, W Huber, S Iacus, R Irizarry, F Leisch, C Li, M Maechler, AJ Rossini, G Sawitzki, C Smith, G Smyth, L Tierney, JY Yang, and J Zhang. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5(10):R80, 2004.

[4] TR Golub, DK Slonim, P Tamayo, C Huard, M Gaasenbeek, JP Mesirov, H Coller, ML Loh, JR Downing, MA Caligiuri, CD Bloomfield, and ES Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–7, Oct 1999.

[5] JA Hartigan and M A Wong. A k-means clustering algorithm. *Applied Statistics*, 28:100–4, 1979.

[6] W Huber, A von Heydebreck, H Sültmann, A Poustka, and M Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 18(Suppl 1):96–104, 2002.

[7] M Kanehisa. Toward pathway engineering: a new database of genetic and molecular pathways. *Sci & Tech Japan*, 59:34–8, 1996.

[8] A von Heydebreck, W Huber, A Poustka, and M Vingron. Identifying splits with clear separation: a new class discovery method for gene expression data. *Bioinformatics*, 17(Suppl 1):S107–14, 2001.

[9] Kaufman L, Rousseeuw PJ (1990) Finding Groups in Data: An Introduction to Cluster Analysis. New York: Wiley.

[10] MacQueen JB (1967) Some methods for classification and analysis of multivariate observations. In: Symposium on Math, Statistics, and Probability. volume 1, pp. 281–97.

[11] Milligan G, Sokol L (1980) A two stage clustering algorithm with robust recovery characteristics. Educational and Psychological Measurement 40:755–9.