Quick start guide for maigesPack – from start to finish

Gustavo H. Esteves

October 9, 2007

Departamento de Matemática e Estatística - Centro de Ciências e Tecnologia - Universidade Estadual da Paraíba, Campina Grande - PB, Brasil,
http://www.maiges.org/en/software/

Contents

1 Overview

This document provides a brief overview of how to use the *maigesPack* package. This package integrates several R packages in a coordinated way to do several microarray data analysis. Some packages that were integrated are from R CRAN and the other came from BioConductor project.

The packages from CRAN are *amap, class, e1071, MASS, R2HTML, rgl* and *som*. From BioC we used the packages *annotate, convert, graph, limma, marray, multtest* and *OLIN*.

To do all data analysis process it is necessary to follow different steps. These are:

- Reading in data.
- Perform simple diagnostic plots to access quality.
- Normalization.
- Following different methods for data analysis according with the problem.

After the two main pre-processing tasks, exploratory analysis and normalization, the next steps in the statistical analysis depend on the biological question for which the microarray experiment was designed. This package integrate some statistical methods to do several types of data analysis.

2 Getting started

To load the `maigesPack` package in your R session, type `library(maigesPack)`. We demonstrate the functionality of this package using a dataset with some observations of gastro-esophageal data. This dataset was analyzed by our group and resulted in a publication of a research article in *Cancer Research*, under volume 65(16), pages 7127–7136 (2005). Following we will show an analysis from start to finish using a piece of this dataset, that is included as part of this package, type `data(gastro)` to load some objects into your R session. Type `?gastro` to get more information about this dataset.

The original data have 4800 spots (aproximately 4400 unique genes) and 172 chips with dye swap, what resulted in a dataset of 86 samples. The subdataset included in this package has 500 spots (representing 486 unique genes) and 40 chips (from 20 observations) between the total 172.

The implementation we did use labels for samples and genes. For the `gastro` dataset we have three important labels for samples and 5 important labels for genes. To samples we have "Sample" that shows an unique identification to each observation, "Tissue"

that shows the tissue classification for each observation (Aeso is Adenocarcinoma from esophagus, Aest is adeno from stomach, Neso is normal esophagus and Nest is normal stomach) and "Type" that shows the general type of the observations (Col is columnar tissue and Sq is squamous tissue). For genes the important labels are "Name", "GeneId", "GeneName", "ClusterId" and "Annot".

To see all sample labels available type `names(gastro@Slabels)`, analogously, type `names(gastro@Glabels)` to see all gene labels available inside the dataset.

3 Loading the dataset

Supposing that the user will load a new dataset, we describe how this can be done. To begin, users must have a gene map file giving all the information necessary to correctly identify genes into the chip, a sample file giving all the information relevant to identify the samples and the data tables generated by the image processing software. If data analysis methods to be used will use gene groups and/or gene networks, users must have this information in other one or two folder(s). The gene map and sample file for the gastro dataset inside this package are available at `/doc/gastro/` directory in the installation folder.

It is important to verify the integrity of data tables, gene map and sample file. Specially, we strongly recommend the user to verify the number of lines from data tables, the layout of the chips used and ordering of the genes. Sometimes we have seen several problems with these characteristics. For the gastro dataset included in this package, we didn't add the numerical data tables because they are very large, having approximately 1.3Mb each.

To start this tutorial open R in any working directory, load the `maigesPack` package and the gastro dataset using the commands given below:

```
> library(maigesPack)
> data(gastro)
```

The object `gastro` loaded above, was created using the function `loadData`. After the check of all data tables, gene map and sample file, the user must use this function to load numerical values into R. The command is given below:

```
> gastro = loadData(fileConf = "load_gastro.conf")
```

the `load_gastro.conf` file specify the parameters needed to load the dataset. Type `?loadData` to see the description of what this file must contain. Pay attention, any error in the execution of this command will be redirected into a file named `load.out`. Once the `gastro` object was created, that is of class `maigesPreRaw`, it is possible to load gene groups and gene networks information using the two commands below, respectively:

```
> gastro = addGeneGrps(gastro, folder = "geneGrps", ext = "txt")
> gastro = addPaths(gastro, folder = "geneNets", ext = "tgf")
```

Pay attention that the two steps above are optional. The user may skip from loading gene groups and networks if he/she don't wanna to use these information during the analysis process. The object generated in this step has class `maigesPreRaw`. The only methods defined to this class of objects is `print, summary` and `show`.

This class of objects is intended to serve as a lobby region into R, where the dataset loaded may be exploratory evaluated using all R graphics functions. Function like `plot, pairs, image,` etc may be used to search by eventual problems into dataset. Also it is possible to do any type of calculation in any data field if the user judge necessary.

The next step of data analysis is to convert the `maigesPreRaw` object named gastro into an object of class `maigesRaw`. This is done with the command below:

```
> gastro.raw = createMaigesRaw(gastro, greenDataField = "Ch1.Mean",
+     greenBackDataField = "Ch1.B.Mean", redDataField = "Ch2.Mean",
+     redBackDataField = "Ch2.B.Mean", flagDataField = "Flags",
+     gLabelGrp = "GeneName", gLabelPath = "GeneName")
```

3.1 Exploratory analysis

From the object `gastro.raw`, that is of class `maigesRaw`, the user can do several exploratory analysis. The function `plot` may be used to show WA plots (that is equivalent to the MA plots, but with W=log2[Test/Ref] instead of M=log2[Red/Green]). The command below do this plot for the first chip and the result is represented at Figure 1:

```
> plot(gastro.raw[, 1], bkgSub = "none")
```

Changing the parameter `bkgSub`, the user change the method of background subtraction, for example, if the user specify `bkgSub="subtract"` the conventional background subtraction is done. You may also do representations for the numerical values into the chip, using the method `image`. See command below, that represents the W values (the W values are represented by default, ie, without any additional parameter, if additional parameters are added, the conventional M values are used) for the first chip that generated the Figure 2:

```
> image(gastro.raw[, 1])

[1] FALSE
NULL
```

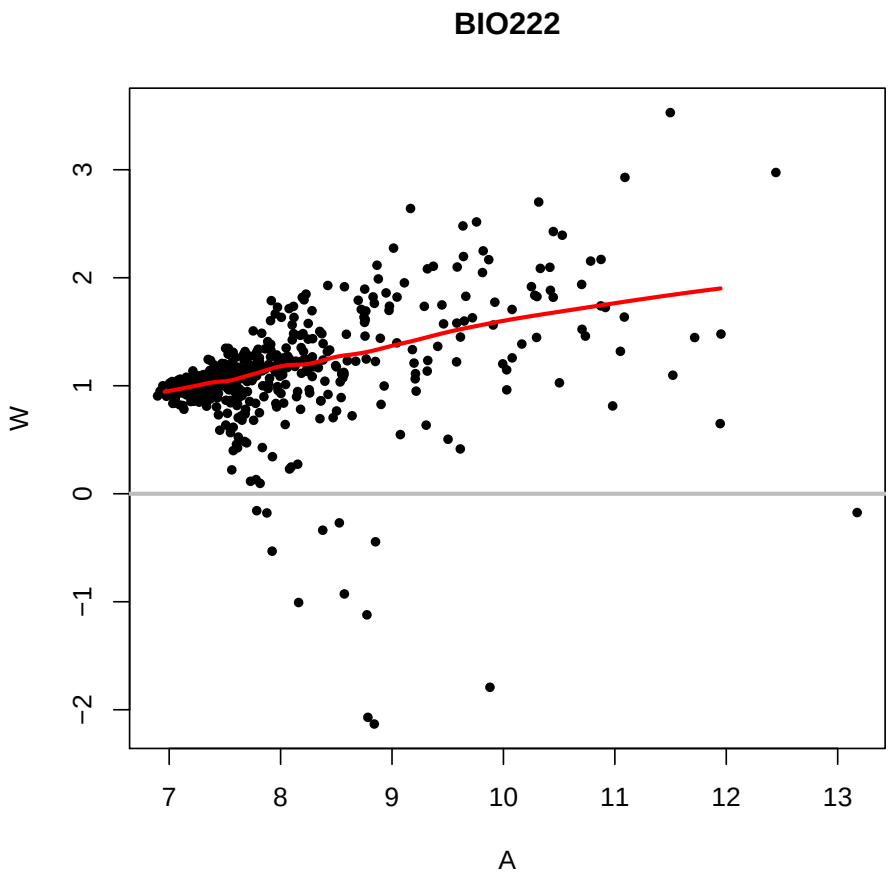

Figure 1: WA plot for the first chip of the gastro data.

To change the W for another value, like A values, use `image(gastro.raw[,1], "maA")`. Take a look at `maImage` method from package *marray* to see additional parameters for this function.

Another type of exploratory graphics is the boxplot. See the commands below:

```
> boxplot(gastro.raw[, 2])
> boxplot(gastro.raw)
```

The user may also do an hierarchical cluster to visualize the quality of the data. Specially when replicates of the observations are done (like the dye swap). Use the command below to construct the hierarchical dendrogram represented at Figure 3.

```
> hierM(gastro.raw, rmGenes = c("BLANK", "DAP", "LYS", "PHE", "Q_GENE",
+     "THR", "TRP"), sLabelID = "Sample", gLabelID = "Name", doHeat = FALSE)
```

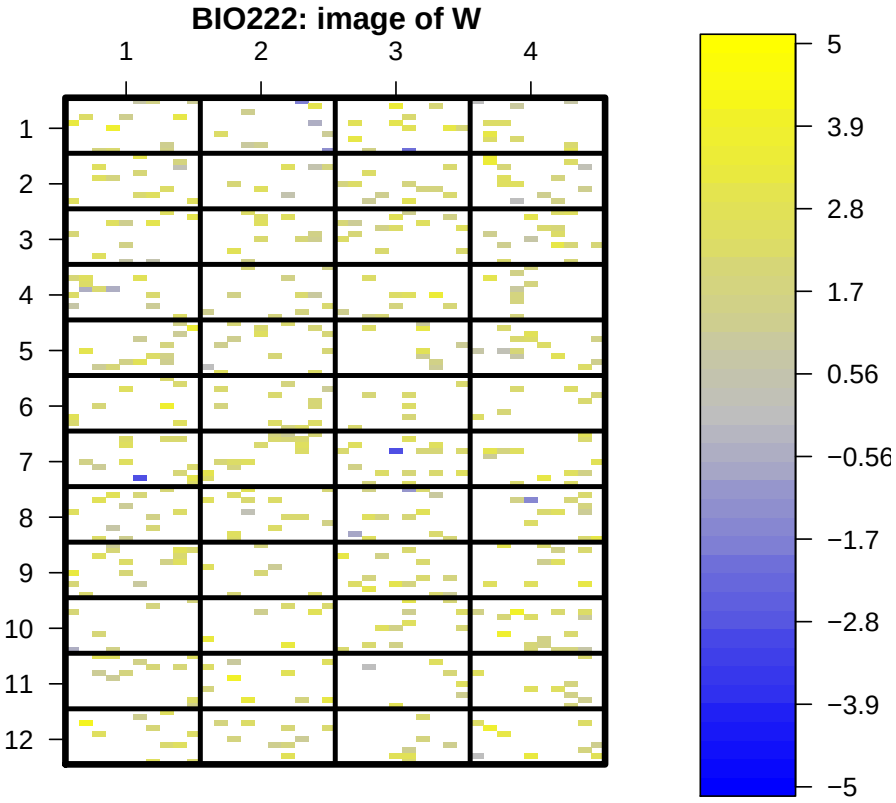

Figure 2: Image for the first chip of the gastro data.

4 Normalizing the dataset

The first step to normalize the data is to select the spots to be used for estimating the normalization factor. In this package this is done by the function `selSpots`. In our example use the command:

```
> gastro.raw2 = selSpots(gastro.raw, sigNoise = 1, rmFlag = NULL,
+     gLabelsID = "Name", remove = list(c("BLANK", "DAP", "LYS",
+         "PHE", "Q_GENE", "THR", "TRP")))
```

This function automatically sets the spots for each chip (column) independently that will be used according with the criteria specified. For example, if the user want to remove spots with signal to noise ratio less than 1.5, specify this value as `sigNoise=1.5`, to remove spots marked with flags 1 and 4, specify `rmFlag=c(1,4)`.

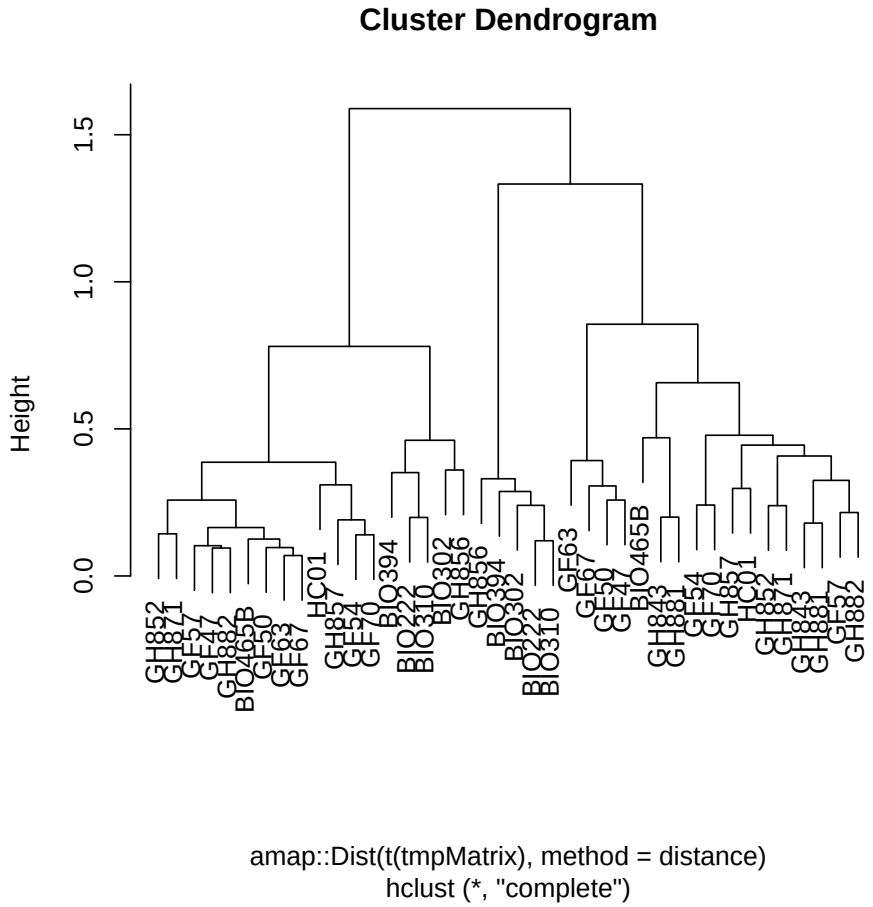

Figure 3: Hierarchical cluster for all observations from raw gastro dataset.

Once this is done, the main function to do the normalization is `normLoc` that uses another function from *limma* package. The most usual normalization method may be applied by the function:

```
> gastro.norm = normLoc(gastro.raw2, span = 0.4, method = "loess")
```

There are another possibilities to do the normalization, like the method OLIN implemented in a package with the same name, and the method that repeats the loess fitting calculating the standard variation and confidence interval. These may be done by using the commands below, but pay attention with them because they are very time consuming. It is interesting to look their arguments.

```
> gastro.norm = normOLIN(gastro.raw2)
> gastro.norm = normRepLoess(gastro.raw2)
```

After the locale normalization the user can use the functions `normScaleMarray` and `normScaleLimma` to do scale adjustment. The first method use a function from *marray* package and the second use a function from *limma* package. To do scale adjustment between print tips use the command below:

```
> gastro.norm = normScaleMarray(gastro.norm, norm = "printTipMAD")
```

To adjust the scale between chips estimating the factor for adjustment by MAD use the command:

```
> gastro.norm = normScaleMarray(gastro.norm, norm = "globalMAD")
```

Limma package offers other possibilities. These were also incorporated into this package. To do the scale adjustment for chips using an estimator slightly different from the MAD, use:

```
> gastro.norm = normScaleLimma(gastro.norm, method = "scale")
```

Another possibility is to do the stabilization of the variance along A values for all chips. But this method must be applied directly over the raw object. This may be done using the command below. This method is also very time consuming.

```
> gastro.norm = normScaleLimma(gastro.raw2, method = "vsn")
```

The normalization functions generates objects of class `maiges`. All the exploratory functions may also be used with this class of objects. As an example, see the Figure 4, produced by the following command:

```
> plot(gastro.norm[, 1])
```

To see the pairing between replicates in dye swap the user can do the hierarchical grouping like what was done in the raw object. That is done by the command below and is represented in Figure 5.

```
> hierM(gastro.norm, rmGenes = c("BLANK", "DAP", "LYS", "PHE",
+     "Q_GENE", "THR", "TRP"), sLabelID = "Sample", gLabelID = "Name",
+     doHeat = FALSE)
```

Sometimes, the sequence (or genes) fixed onto spots may be replicated. The same thing may happen with the observations (the most common is dye swap). So depending on the statistical models used, the user must have to resume the data both for spots and biological observations. These may be done by using the function:

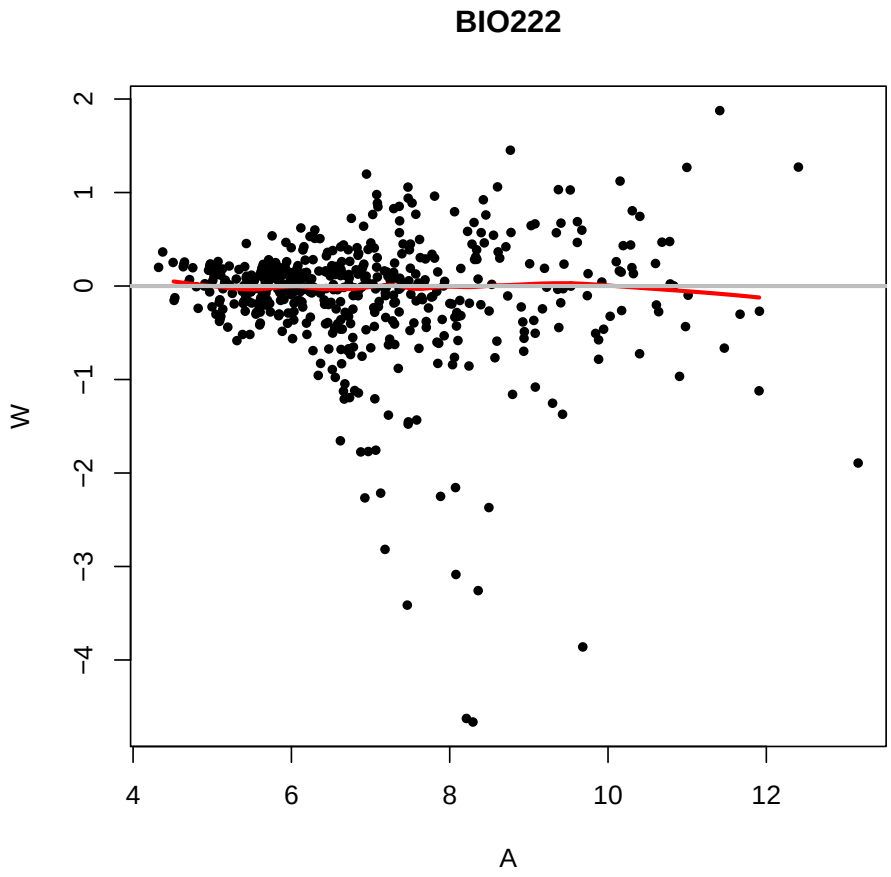

Figure 4: WA plot for the first chip of the gastro data after normalization.

```
> gastro.summ = summarizeReplicates(gastro.norm, gLabelID = "GeneName",
+     sLabelID = "Sample", funcS = "mean", funcG = "median", keepEmpty = FALSE,
+     rmBad = FALSE)
```

If the user needs to resume only spots or samples, simply specify `funcS=NULL` or `funcG=NULL`. If both of them are `NULL` no summary are done.

5 Some data analysis methods

After the normalization, or pre-processing step, the user may use several statistical methods for data analysis. In this package we integrated some methods that were already available into R or BioConductor. Between these methods we have cluster algorithms, differential gene expression and classifiers search, functional classification of gene groups

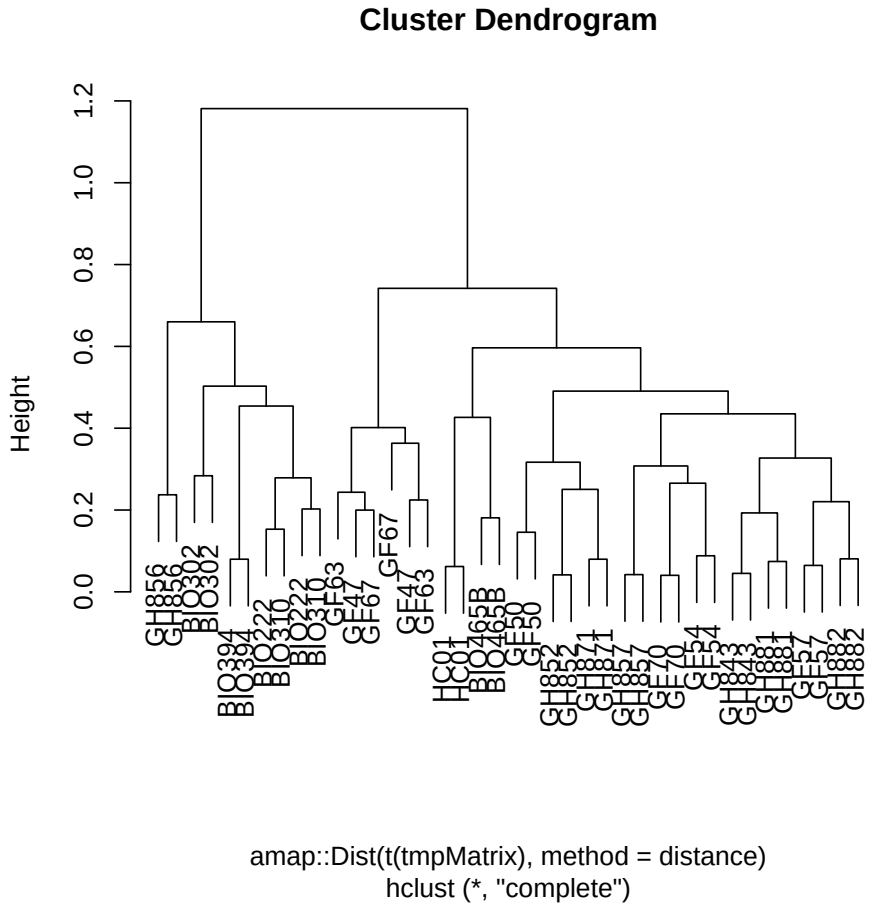

Figure 5: Hierarchical cluster for normalized data of all observations from gastro data.

or gene networks and construction of relevance networks.

5.1 Differentially expressed genes (DE genes)

We implemented three main methods to search by DE genes: `deGenes2by2Ttest`, `deGenes2by2Wilcox` and `deGenes2by2BootT`. The first one supposes that the data are normally distributed and uses the t statistic to do the calculations. The function may be used as in the following example:

```
> gastro.ttest = deGenes2by2Ttest(gastro.summ, sLabelID = "Type")
> gastro.ttest

Object of class maigesDEcluster generated by Tue Oct  9 15:09:58 2007,
 using R version 2.6.0 (2007-10-03).
```

```
Classifying 486 genes as Differentialy Expressed (DE)
 i  1 test(s).

The method used was T test.
```

In cases were the data don't seen to be normally distributed. The user may use the other two methods that are non-parametric. The method `deGenes2by2BootT` is very time consuming. See two examples of this methods below.

```
> gastro.wilcox = deGenes2by2Wilcox(gastro.summ, sLabelID = "Type")
> gastro.boot = deGenes2by2BootT(gastro.summ, sLabelID = "Type")
```

Once the analysis was done. The user can see volcano plots and save HTML (or CSV) tables with the results using the commands below, respectively.

```
> plot(gastro.ttest)
> tablesDE(gastro.ttest)
```

It is also possible to do cluster analysis selecting DE genes according with the p-values of the tests. The functions `hierMde`, `somMde` and `kmeansMde` do cluster analysis using hierarchical, SOM and k-means algorithms, respectively. To do the SOM cluster with 2 groups using 20 most differentially expressed genes (adjusting p-values by FDR) use the command below, the result is in the Figure 6

```
> somMde(gastro.ttest, sLabelID = "Type", adjP = "BH", nDEgenes = 20,
+     xdim = 2, ydim = 1, topol = "rect")
```

Similarly, it is possible to do cluster analysis directly for objects of class `maiges`, using the functions `hierM`, `somM` and `kmeansM`, but without selecting differentially expressed genes.

All the methods presented above are applicable to compare two distinct biological sample types. When there are more than two types, it is possible to use ANOVA models. We also implemented a method to do this kind of analysis. But first the user must run another method to construct design and contrasts matrices. To construct these matrices for an ANOVA model for "Tissue" factor (note that this factor has 4 sample types, type `getLabels(gastro.summ, "Type")` to see them), use the command:

```
> gastro.ANOVA = designANOVA(gastro.summ, factors = "Tissue")
```

And them, to fit the model and to do the F test use the command:

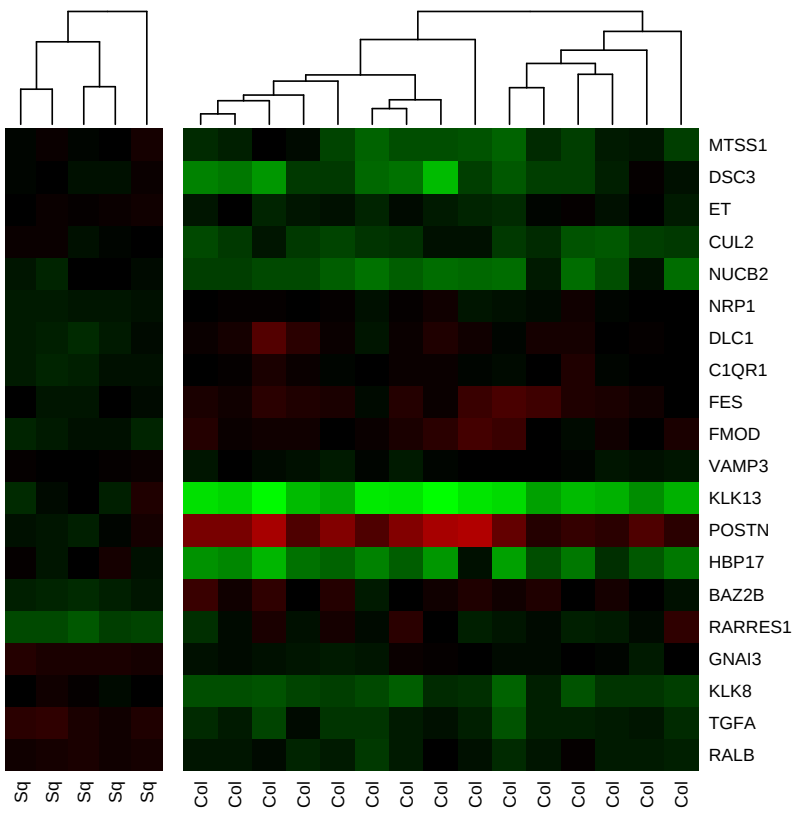

Figure 6: SOM cluster with 2 groups using 20 most DE genes.

```
> gastro.ANOVAfit = deGenesANOVA(gastro.ANOVA, retF = TRUE)
> gastro.ANOVAfit

Object of class maigesDEcluster generated by Tue Oct  9 15:09:59 2007,
 using R version 2.6.0 (2007-10-03).

Classifying 486 genes as Differentialy Expressed (DE)
 i  1 test(s).

The method used was ANOVA - F test.
```

If the user want to do the individual t tests use (this is the default):

```
> gastro.ANOVAfit = deGenesANOVA(gastro.ANOVA, retF = FALSE)
```

Another function that may be useful, specially for ANOVA analysis is `boxplot`. This type of plot may help in identify the tissues where some gene presented alteration. For example the gene FUBP1, presents significantly alteration in the F test done above. Using the command below it is possible to see in what tissue this gene is altered. The result is illustrated in the Figure 7. The method `boxplot` also works with objects of class `maiges`.

```
> boxplot(gastro.ANOVAfit, name = "KLK13", gLabelID = "GeneName",
+     sLabelID = "Tissue")
```

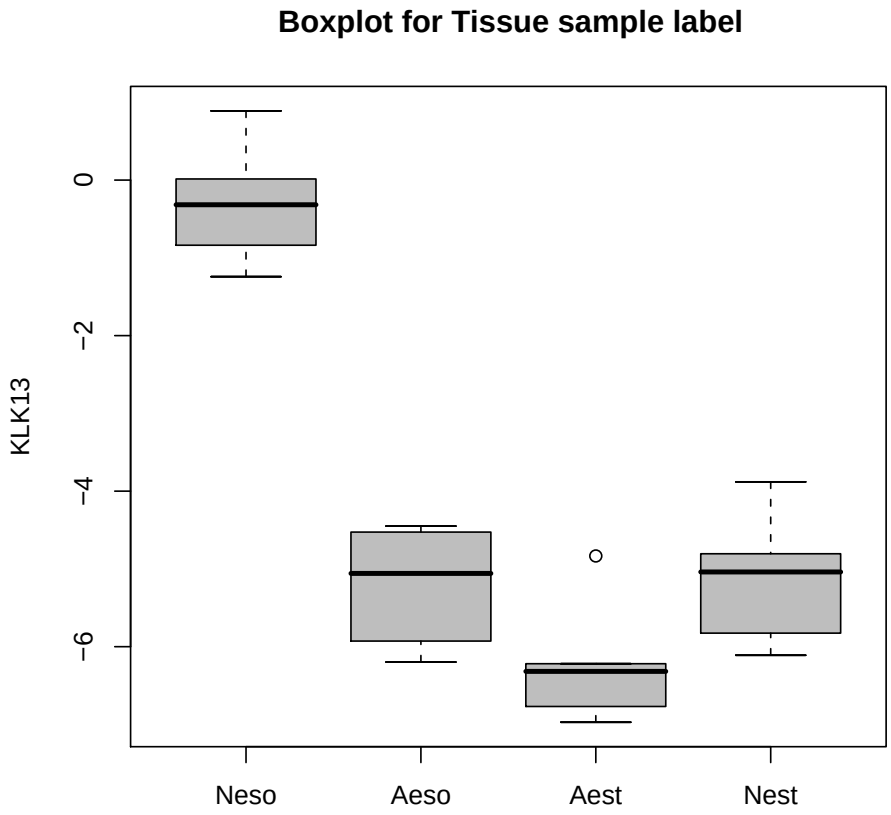

Figure 7: Boxplot for the gene FUBP1 separating by tissues.

The object resulted from this fitting are also of class `maigesDE` (or `maigesDEcluster`). So, the commands for plot, to save tables and to construct clusters, presented above also works here. Pay attention that functions to do cluster analysis only work for objects of classes `maigesDEcluster`.

5.2 Discrimination analysis

Another type of data analysis that were included into this package is the discrimination (or classification) analysis. This kind of analysis search by groups of few genes that can distinguish between different sample types. To evaluate the quality of the groups, we use the cross validation method. This was implemented in the functions `classifyLDA`, `classifySVM` and `classifyKNN` for methods of Fisher's linear discriminant analysis, SVM and k-neighborhood, respectively.

It is possible to search exhaustively for all genes from the dataset. But this is very time consuming. So, we added the possibility to search for classifiers in gene groups or networks. To search by trios of genes that classify the two types ("Col" and "Sq") using the Fisher's method, for the seventh (cell cycle arrest, gene ontology code GO0007050) gene group into `GeneGrps` slot from gastro.summ object use the command below:

```
> gastro.class = classifyLDA(gastro.summ, sLabelID = "Type", gNameID = "GeneName",
+     nGenes = 3, geneGrp = 6)
> gastro.class

Object of class maigesClass generated by Tue Oct  9 15:10:01 2007,
 using R version 2.6.0 (2007-10-03).

56 classifiers of 3 genes, ordered by CV value.

The method used was Fisher LDA by exaustive search.
```

If the user want to find groups of three genes that discriminate the tissues classified as normal ("Neso" and "Nest") versus tumor ("Aeso" and "Aest"), it is possible to use the command below:

```
> gastro.class = classifyLDA(gastro.summ, sLabelID = "Tissue",
+     gNameID = "GeneName", nGenes = 3, geneGrp = 6, facToClass = list(Norm = c("Neso",
+         "Nest"), Ade = c("Aeso", "Aest")))
```

To visualize or save tables (HTML or CSV) for the classifiers it is possible to use the two commands below:

```
> plot(gastro.class, idx = 1)
> tableClass(gastro.class)
```

As the exhaustive search is very time consuming for large groups of genes there is another possibility implemented, the search and choose method. The use is similar with the previous and the functions are `classifyLDAsc`, `classifySVMsc` and `classifyKNNsc`.

5.3 Functional classification of gene groups

Another data analysis method implemented in this package is the functional classification of gene groups (or modules), proposed by Segal et al (Nature Genetics, vol: 36(10), pp: 1090-1098, 2004). This method basically searches for groups of genes that present numbers of differentially expressed genes greater than the expected by chance.

The user can do this kind of analysis for all gene groups loaded into the dataset (this information is stored in the slot `GeneGrps` from objects of classes `maigesPreRaw`, `maigesRaw` and `maiges`) for sample label "Tissue" using the command below:

```
> gastro.mod = activeMod(gastro.summ, sLabelID = "Tissue", cutExp = 1,
+     cutPhiper = 0.05)
```

To plot the results, as an image (like a heatmap), for the individual observations the user can use the command:

```
> plot(gastro.mod, "S", margins = c(15, 3))
```

For each tissue the user can use the command below. The result is represented in the Figure 8.

```
> plot(gastro.mod, "C", margins = c(23, 5))
```

The function also calculate an score (with significance levels, p-values) given the concordance of the gene and group classification. These values can be saved, as HTML tables, for each sample type using the command `activeModScoreHTML`.

5.4 Relevance networks

To evaluate the interaction between genes into groups it is possible to use reverse engineering strategies. One of this techniques was proposed by Butte et al (PNAS, vol: 97(22), pp:12182-12186, 2000), where we estimate the correlation values between all pairs of genes from a given gene group and test the significance of this value under the hypothesis of null correlation. Then, we select the pairs with correlations values significantly different from zero using some p-value cutoff. To construct relevance networks for normal esophagus tissue ("Neso") in the seventh gene group, use this command:

```
> gastro.net = relNetworkB(gastro.summ, sLabelID = "Tissue", samples = "Neso",
+     geneGrp = 1)
```

To visualize the results it is possible to do an image of the correlation values for all pairs of genes or a graph representing that ones with p-values less than `cutPval`. This can be done by the two commands:

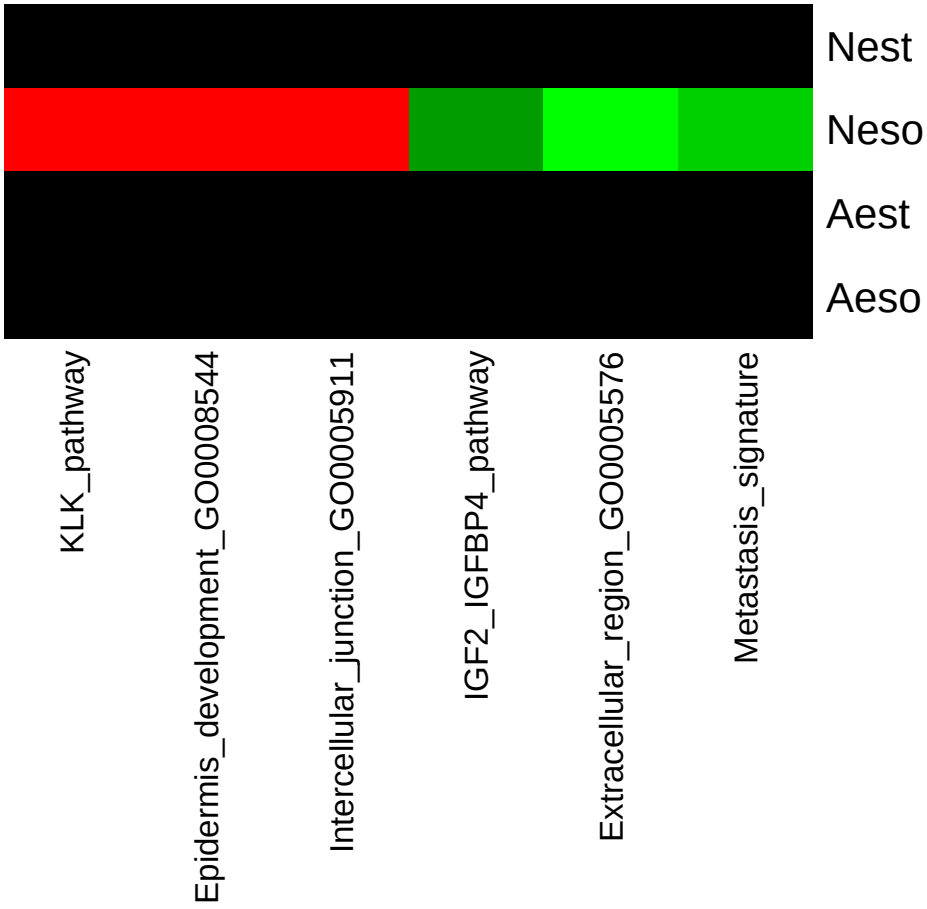

Figure 8: Result from functional classification of gene groups for "Tissue" label.

```
> image(gastro.net)
> plot(gastro.net, cutPval = 0.05)
```

However, the method proposed by Butte et al don't turn possible to compare quantitatively the results obtained in two different sample types. So, we implemented another method that construct relevance networks with pairs of genes presenting altered correlation values between two sample types. This is done by a Fisher's Z transformation. To find pairs of genes that present altered correlation values between normal esophagus and adenocarcinoma from esophagus, use the command:

```
> gastro.net2 = relNetworkM(gastro.summ, sLabelID = "Tissue", samples = list(Neso = "Neso",
+     Aeso = "Aeso"), geneGrp = 7)
```

Also, it is possible to use the commands `image` and `plot` to visualize the results. See the commands:

```
> image(gastro.net2)
> plot(gastro.net2, cutPval = 0.05)
```

It is possible to represent the gene expression profiles in the two sample types tested representing the regression line between them, see the command:

```
> plotGenePair(gastro.net2, "KLK13", "EVPL")
```

5.5 Functional classification of gene networks

Finally, we noted that any of the last two data analysis methods were able to evaluate the activation profiles of gene networks. We proposed and implemented an statistical method to measure the significance of gene networks activation. This is done by a statistic that have a known gamma distribution, and them, we test if the value of this statistic for each sample type can happen randomly. To classify gene networks (stored in the slot `Paths` from objects of classes `maigesPreRaw`, `maigesRaw` and `maiges`) for "Tissue" label the user can use the command:

```
> gastro.net = activeNet(gastro.summ, sLabelID = "Tissue")
```

To see an image representing the statistic (scores) for each tissue, use the command below. The image is represented in Figure 9. The color varies from black (small score) to blue (high score).

```
> plot(gastro.net, type = "score", margins = c(21, 5))
```

Similarly, the user can see an image representing the p-values of the tests for each tissue, this can be done using the command below. The image is represented in Figure 10. As in the previous figure the color varies from black (less significance) to blue (high significance).

```
> plot(gastro.net, type = "p-value", margins = c(21, 5))
```

It is also possible to save the score and p-values, as HTML tables, for each sample type using the command `activeNetScoreHTML`.

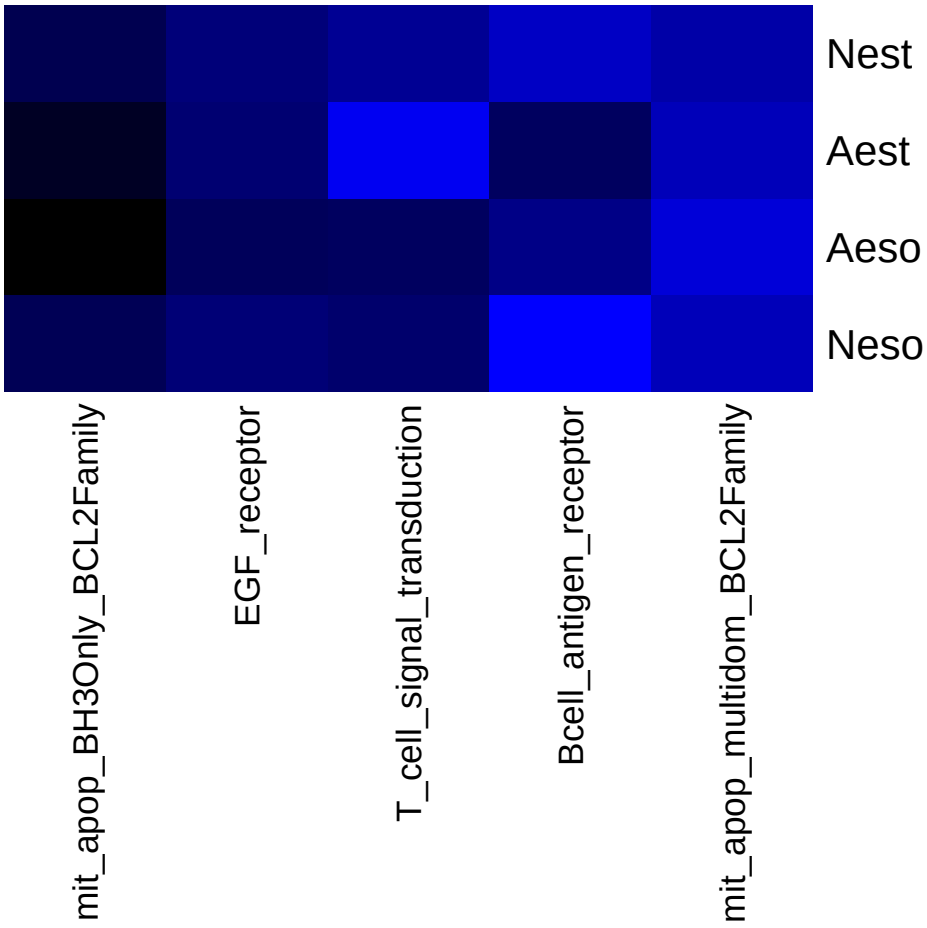

Figure 9: Image representing the scores (statistics) from functional classification of gene networks for "Tissue" label.

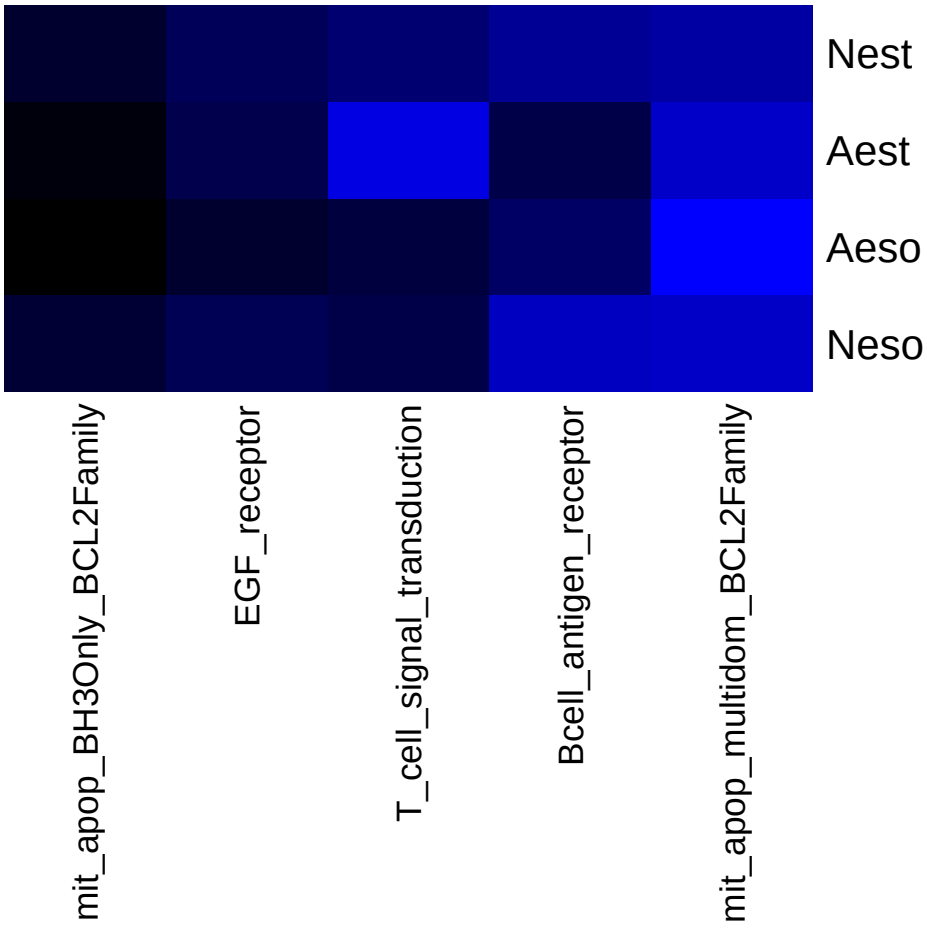

Figure 10: Image representing the p-values from functional classification of gene networks for "Tissue" label.