CSFA 1.1.0 - Vignette

Ewoud De Troyer

1 Introduction

One of the many challenges in today's omics data is the goal of connecting those compounds/molecules/samples together which have similar properties by gene expression. Techniques like this allow the discovery of new molecule properties by connecting their signatures with those derived from already well-known ones.
Papers such as Lamb et al. (2006) and Zhang and Gant (2008) both already took up the challenge of dealing with this problem. In Lamb et al. (2006), a reference collection of gene expression profiles from human cells treated with bioactive small molecules was created in order to design a systematic approach to discover these functional connections. Query signatures of interest (e.g. from a study) are compared to the reference profiles and connectivity scores are computed. While their approach achieved a good degree of succes, it was unable to measure statistical significance. This is where the paper from Zhang and Gant (2008) continued for example. Their paper offers a more principled statistical procedure to test connections between the compounds which allows the valuation of statistical significance.

The CSFA package accompanies the submitted paper "Connectivity Mapping: A Multivariate Approach Using Multiple Factor Analysis" by De Troyer,E. et al. (2018), which proposes the usage of *factor analysis* methods (Principal Component Analysis (=PCA), (Sparse) Multiple Factor Analysis (MFA) (Abdi et al., 2013) (or FABIA (Factor Analysis for Bicluster Acquisition) (Hochreiter et al., 2010))) to derive the connectivity between compounds. Using these methods, not only do you obtain information about the connectivity between the compounds, you also get information about which genes are responsible for guiding this connectivity.
Further instead of computing a pairwise correlation/connection score between the compounds, now the entire available data is being used to look for dominant structures on both dimenstions. This is very similar to try to discover biclusters in the data. Consequently, it is not necessary anymore to decide upon a cut-off for up- and downregulated genes since you will be using all the genes to do the factor analysis.
In our new proposed method, we start from a small set of query samples (or single) which are known to be similar on a gene expression level. These are compared with a larger set of reference samples in order to try to discover samples or compounds similar to the query set. These two sets or matrices (query and reference) share a common dimension, namely the genes.

$$\overbrace{\phantom{\mathbf{X}_1}}^{Query\ Samples}\ \overbrace{\phantom{\mathbf{X}_2}}^{Reference\ Samples}$$
$$\left[\begin{array}{c|c} \mathbf{X}_1 & \mathbf{X}_2 \end{array}\right] \quad g \text{ genes} \qquad (1)$$
$$n \text{ samples}$$

Finally in order to easily compare these methods with the Zhang and Gant Score, CSFA also includes an implementation of this algorithm together with the ability to compare the scores with the FA scores.

2 Data

In order to showcase the functionality of CSFA, some simulated microarray data will be used. The data contains 1000 genes and 341 compounds of which 6 will be used as query signatures. The remaining query signatures consist out of 5 strongly positive connected compounds, 20 weakly positive connected compounds, 10 strongly negative connected compounds and 300 compounds which are not connected at all.

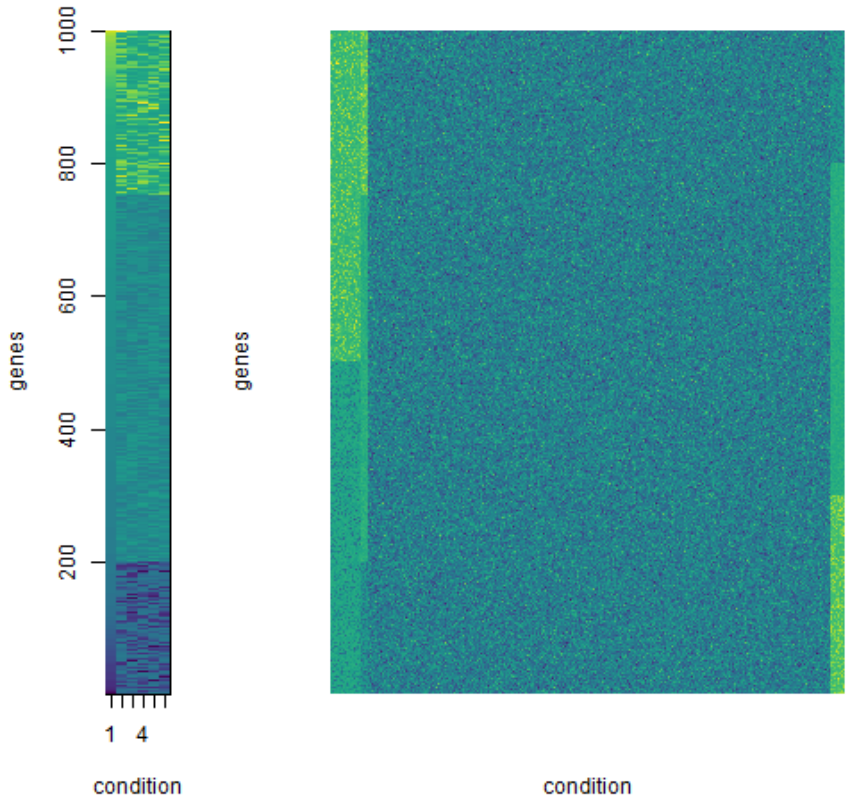

Figure 1: Heatmap of Query and Reference Matrix

3 Example CS Analysis

Start by first loading both the CSFA library and the example data available in the package. The simulated data is split up in the query and reference matrix.

```
library(CSFA)
data("dataSIM",package="CSFA")

querMat <- dataSIM[,c(1:6)]
refMat <- dataSIM[,-c(1:6)]
```

Next, the Connectivity Scores from Zhang and Gant, MFA and FABIA will be computed with the package. The last two methods will also provide scores for the genes involved in the structure.
More details about the connectivity and gene scores as well as the decision making of which component to look at can be be found in the submitted paper.

3.1 Zhang and Gant

The Zhang and Gant scores are computed with the default parameters. This means all the genes will be used (no cut-off) and the reference signature will be considered as an ordered signature. Also no permutation will be applied by default.
Note that for the vignette, which is a sweave document, we use the "sweave" `plot.type`. Normally you would be using either "device" or "pdf".

```
out_ZG <- CSanalysis(querMat,refMat,"CSzhang",plot.type="sweave")
```

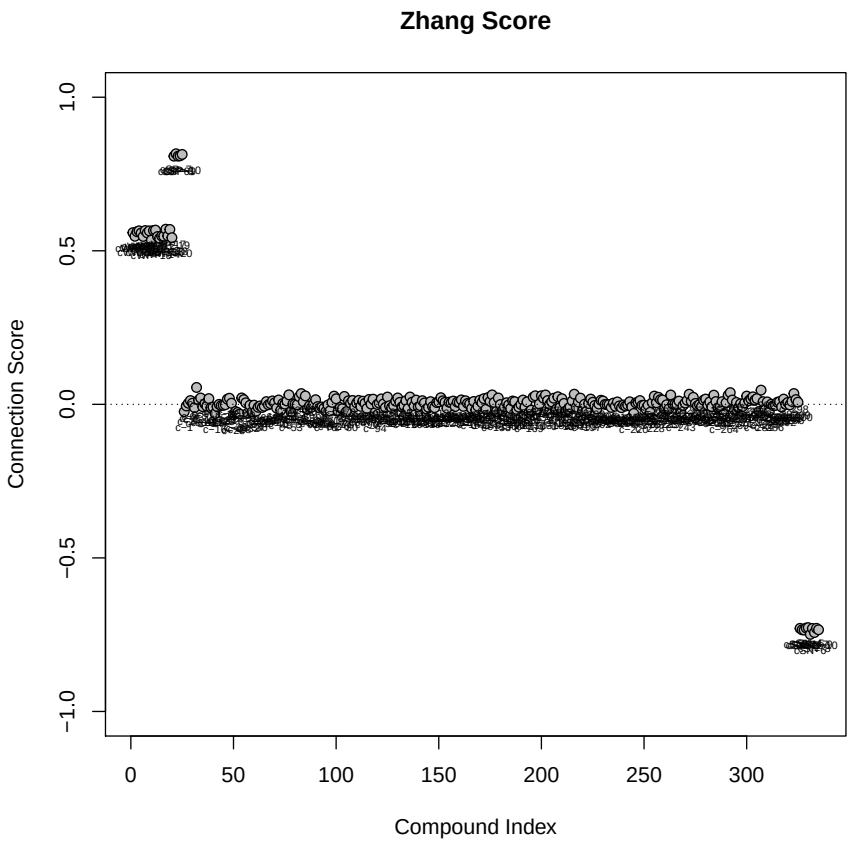

Figure 2: CSanalysis Graphs for CSzhang

The connectivity scores can be found in the `out_ZG` object. Figure 2 clearly shows the positive (weak and strong) and negative connected compounds.

3.2 MFA

The next CS analysis which is applied is the one using Multiple Factor Analysis (MFA) by setting the `type` to `"CSmfa"`. Three of the available plots were chosen, namely the Query Loadings, Compound Loadings (Connectivity Scores), Gene Scores and Compound Profiles (`which=c(2,3,4,7)`).
Note that in the R-code we already preselected which component to investigate with `component.plot`. Further we also already decided which columns of the reference matrix we would like to draw in the compound profiles graph with `column.interest`. Indices 1, 2 and 3 coincide with 3 weakly positive connected compounds.

However, if you are not sure beforehand what you want to investigate, you can also decide upon these parameters interactively on the fly. To do this simply set these parameters to `NULL` or leave them out.
To determine `component.plot`, you will be able to to click on the factors you want to observe in the *"Loadings for Query..."* plot. This graph will be your main guideline on which factor is capturing the structure of your query set of signatures. As shown in Figure 3 below, this is clearly the first factor.
Next, in order to drawn compound profiles, set `profile.type` to `"cmpd"`. The `column.interest` parameter for this plot can also be chosen in the *"Compound Loadings"* plot (instead of simply providing it to `CSanalysis` beforehand). You can left-click on multiple compounds you wish to draw in the compound profiles graph (and right-click to stop the selection procedure).

```
out_MFA <- CSanalysis(querMat,refMat,"CSmfa",plot.type="sweave",which=c(2,3,4,7),
                profile.type="cmpd",gene.thresP=2.3,gene.thresN=-2.3,component.plot=1,
                column.interest=c(21,22,23))
```

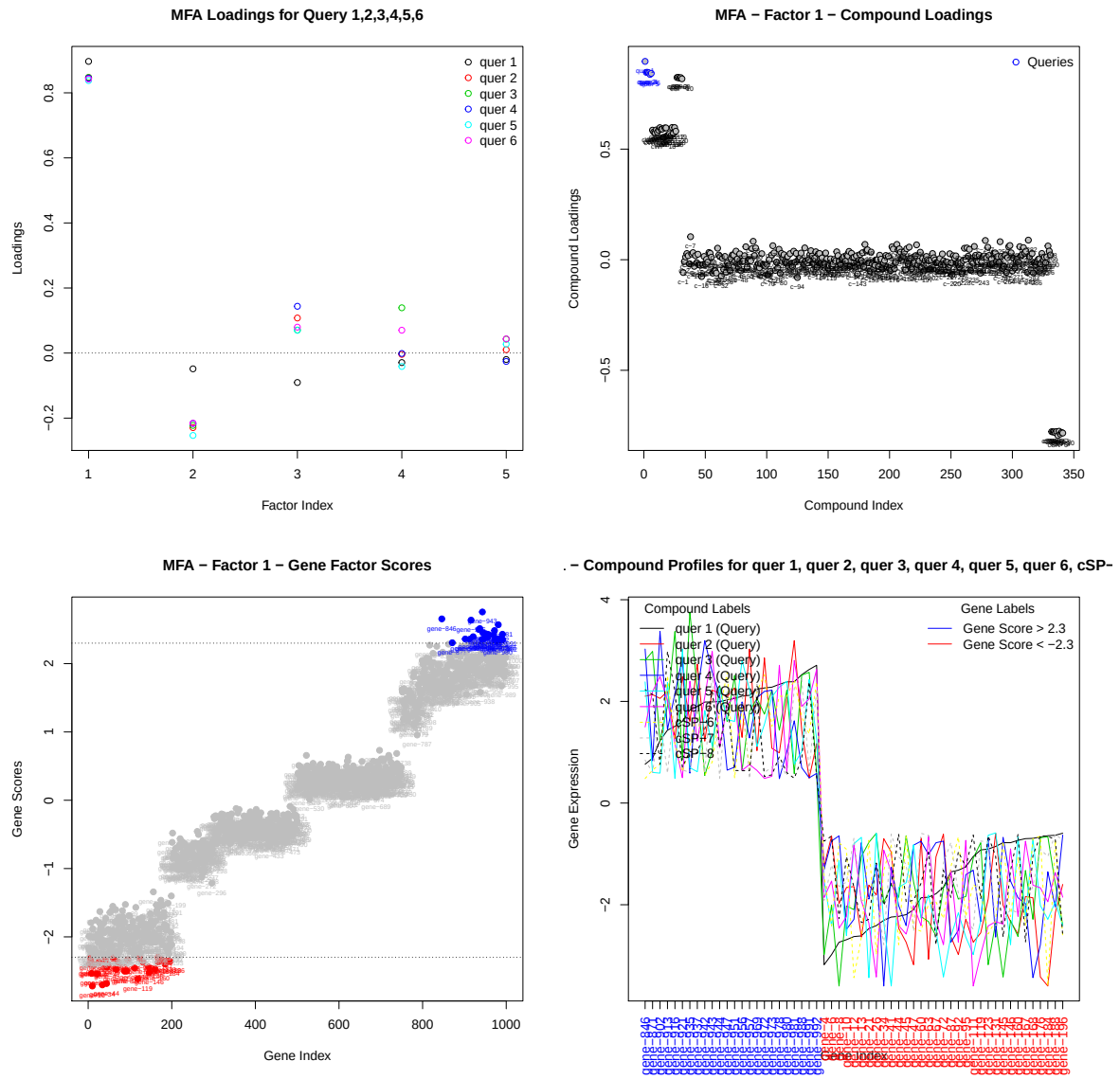

Figure 3: CSanalysis Graphs for CSmfa

Just like in the Zhang and Gant plot, we again see that the simulated positive and negative connected compounds are appearing in the Compound Loadings plot. However now we also get a plot showing the scores of the genes involved in the structure of the first factor in the MFA analysis.

We can also reuse the `CSanalysis` function to draw the same or additional plots without re-computing the factor analysis. This is done through the `result.available` parameter. Here in Figure 4, the Connectivity Score Ranks are shown.

```
out_MFA <- CSanalysis(querMat,refMat,"CSmfa",plot.type="sweave",which=c(5),
                component.plot=1,column.interest=c(1,2,3),result.available=out_MFA)
```

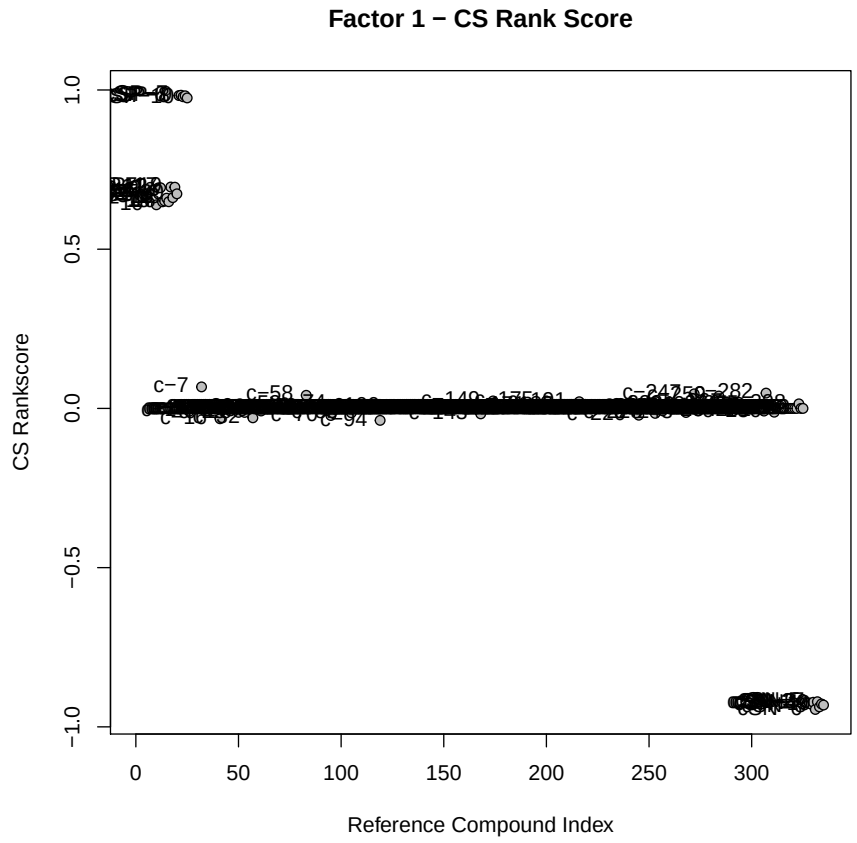

Figure 4: CSanalysis Graphs for CSmfa

Another example would be to draw gene profiles. Now alongside the manual or interactive selected `column.interest`, you can also manual select which genes should be used with `row.interest`. If not provided this is also done interactively in the gene score plot. In this graph, the x-axis contains all compounds, starting with the query and selected ones. The others are the ordered in decreasing CScore.

```
out_MFA <- CSanalysis(querMat,refMat,"CSmfa",plot.type="sweave",which=c(7),
                profile.type="gene",component.plot=1,column.interest=c(1,2,3),
                row.interest=c(846,871,4,6),result.available=out_MFA)
```

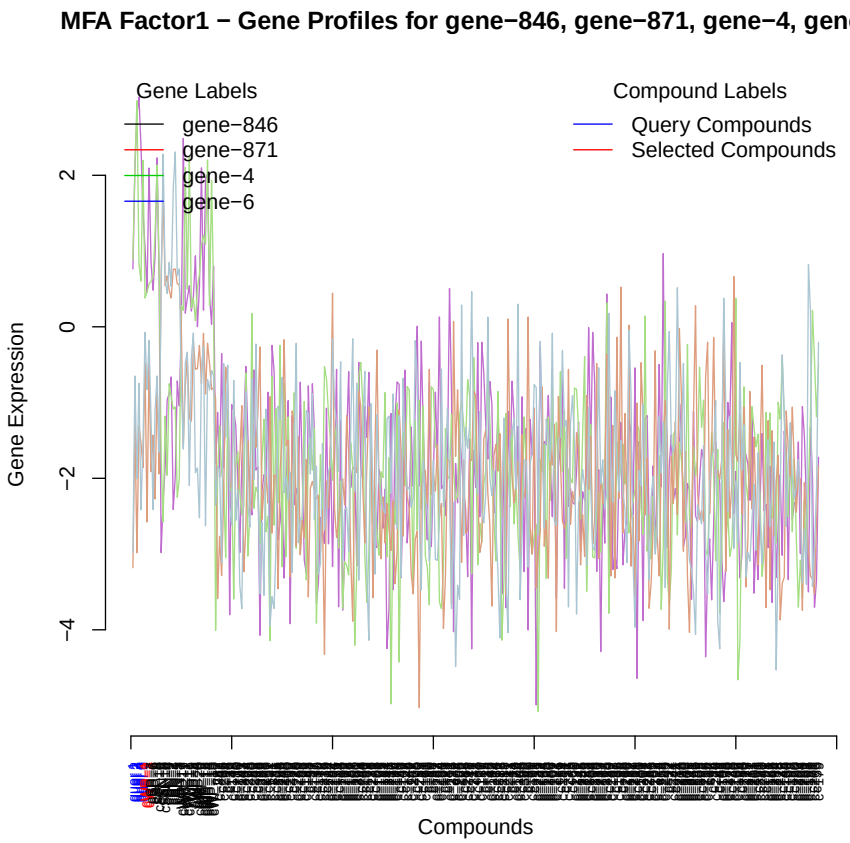

Figure 5: CSanalysis Graphs for CSmfa

3.3 FABIA

The last analysis is done with FABIA, Factor Analysis for Bicluster Acquisition (`type="CSfabia"`). We will only select 2 plots this time, namely the query loadings and compound loadings (`which=c(2,5)`). However in contrary with the MFA anaylsis, we select 2 components for this analysis. Based on the query loadings we decide to select bicluster 1 and 2 (`component.plot=c(1,2)`).
This time we also do some manual coloring of the columns to highlight some strongly connected compounds with `color.columns`. We start by making a vector of length 341 (column dimension of example data) and fill it with the color black. Next we fill in the color blue for the 6 query compounds and red for 3 of the strongly positive connected compounds. We also change the legend according to this coloring.

Note that we have also set a seed just before the FABIA analysis in order to have a reproducible result.

```
color.columns <- rep("black",dim(dataSIM)[2])
color.columns[1:6] <- "blue"
color.columns[c(29,30,31)] <- "red"

set.seed(8956)
out_FABIA <- CSanalysis(refMat,querMat,"CSfabia",plot.type="sweave",which=c(2,5),
                color.columns=color.columns,
                legend.names=c("Queries","SP Connected"),
                legend.cols=c("blue","red"), component.plot=c(1,2),
                gene.thresP=2,gene.thresN=-2)
```

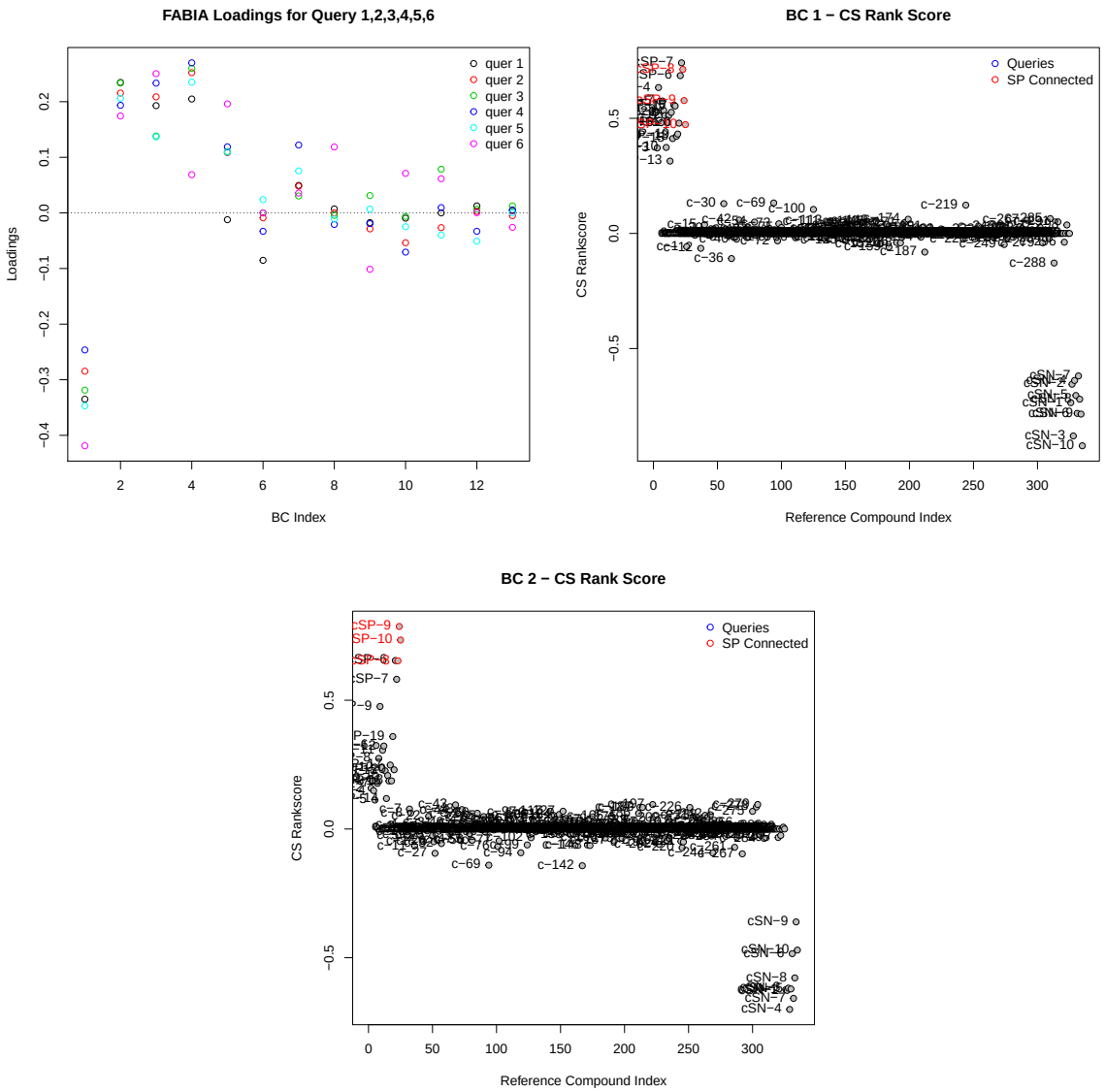

Figure 6: CSanalysis Graphs for CSfabia

The results in Figure 6 are very comparable with the Zhang and MFA graphs.

4 Example CS permutation

The `CSFA` package also contains a function called `CSpermute`. With this function it is possible to compute p-values through permutation for the MFA and Zhang & Gant results.
These results will be added to the `CS` slot of both the MFA and ZG results. More information is also entered in the `permutation.object` slot.

First, let us apply the permutation on the MFA and ZG result without plotting any plots just yet by putting `which` to `c()`. The number of permutations was chosen to only be 250 in this case. Further, the p-values are also adjusted for multiplicity by setting a value for `method.adjust` different than `"none"`.
It is also possible to parallelise (`snowFT`) the permutation process by setting `MultiCores` to `TRUE`. The parameters `MultiCores.number` and `MultiCores.seed` respectively control the number of cores and seed you would like to use. The former defaults to the maximum total of available physical cores.

Note: or MFA, `CSpermute` should *only* be used to compute the p-values of the Component in which the structure (loadings) of the queries is the strongest.

```
out_MFA <- CSpermute(querMat,refMat,CSresult=out_MFA,B=250,method.adjust="BH",
                which=c(),verbose=FALSE,MultiCores=TRUE)
out_ZG <- CSpermute(querMat,refMat,CSresult=out_ZG,B=250,method.adjust="BH",
                which=c(),verbose=FALSE,MultiCores=TRUE)
```

```
head(out_MFA@CS[[1]]$CS.ref)
```

```
##       CLoadings   CLpvalues CLpvalues.adjusted CRankScores   CRpvalues
## cWP-1 0.5858096 0.003984064         0.03813318   0.6800244 0.003984064
## cWP-2 0.5771857 0.003984064         0.03813318   0.6690638 0.003984064
## cWP-3 0.5739775 0.003984064         0.03813318   0.6649862 0.003984064
## cWP-4 0.5867910 0.003984064         0.03813318   0.6812717 0.003984064
## cWP-5 0.5770335 0.003984064         0.03813318   0.6688703 0.003984064
## cWP-6 0.5852554 0.003984064         0.03813318   0.6793200 0.003984064
##       CRpvalues.adjusted CLrank CLabsrank CRrank CRabsrank
## cWP-1         0.03813318     14        24     14        24
## cWP-2         0.03813318     17        27     17        27
## cWP-3         0.03813318     19        29     19        29
## cWP-4         0.03813318     13        23     13        23
## cWP-5         0.03813318     18        28     18        28
## cWP-6         0.03813318     15        25     15        25
```

```
head(out_ZG@CS$CS.ref)
```

```
##         ZGscore     pvalues pvalues.adjusted ZGrank ZGabsrank
## cWP-1 0.5590061 0.003984064       0.03707393     14        24
## cWP-2 0.5473340 0.003984064       0.03707393     20        30
## cWP-3 0.5621351 0.003984064       0.03707393     13        23
## cWP-4 0.5649765 0.003984064       0.03707393     11        21
## cWP-5 0.5578107 0.003984064       0.03707393     16        26
## cWP-6 0.5468158 0.003984064       0.03707393     21        31
```

Next, we can actually re-use the updated `out_MFA` and `out_ZG` in `CSpermute`. As long as the number of permutations (`B`) is not changed, the permutation will not need to computed all over again. This means you can plot the available graphs (which: 1, volcano plot for CLoadings ; 2, CLoadings compound distribution histogram under null hypothesis with p-value), as many times as needed (plot 3 and 4 are the same as 1 and 2, but for CRankingScores). The parameter `cmpd.hist` decides which compounds should be used for the second type of plot. If this parameter is not given (`NULL`), you can interactively choose them on the volcano plot by left-clicking on them (and right-click to stop). In the code below, we plot both type of graphs for the MFA result with a pre-determined `cmpd.hist`.

```
out_MFA <- CSpermute(querMat,refMat,out_MFA,B=250,method.adjust="BH",
                which=c(1,2),cmpd.hist=c(23,99),plot.type="sweave")
```

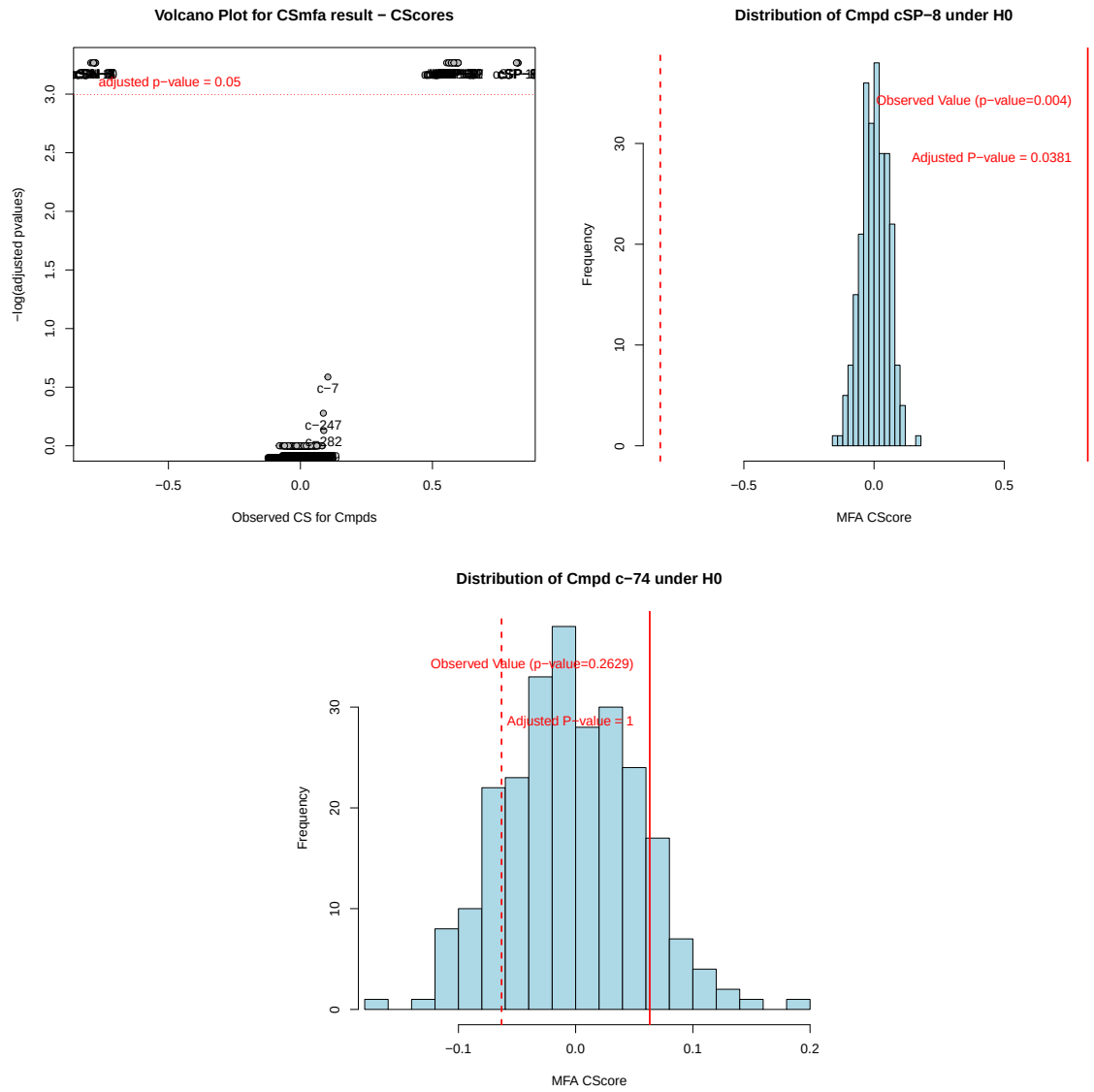

Figure 7: CSpermute graphs for MFA result

```
out_MFA <- CSpermute(querMat,refMat,out_MFA,B=250,method.adjust="BH",
                     which=c(3,4),cmpd.hist=c(23,99),plot.type="sweave")
```

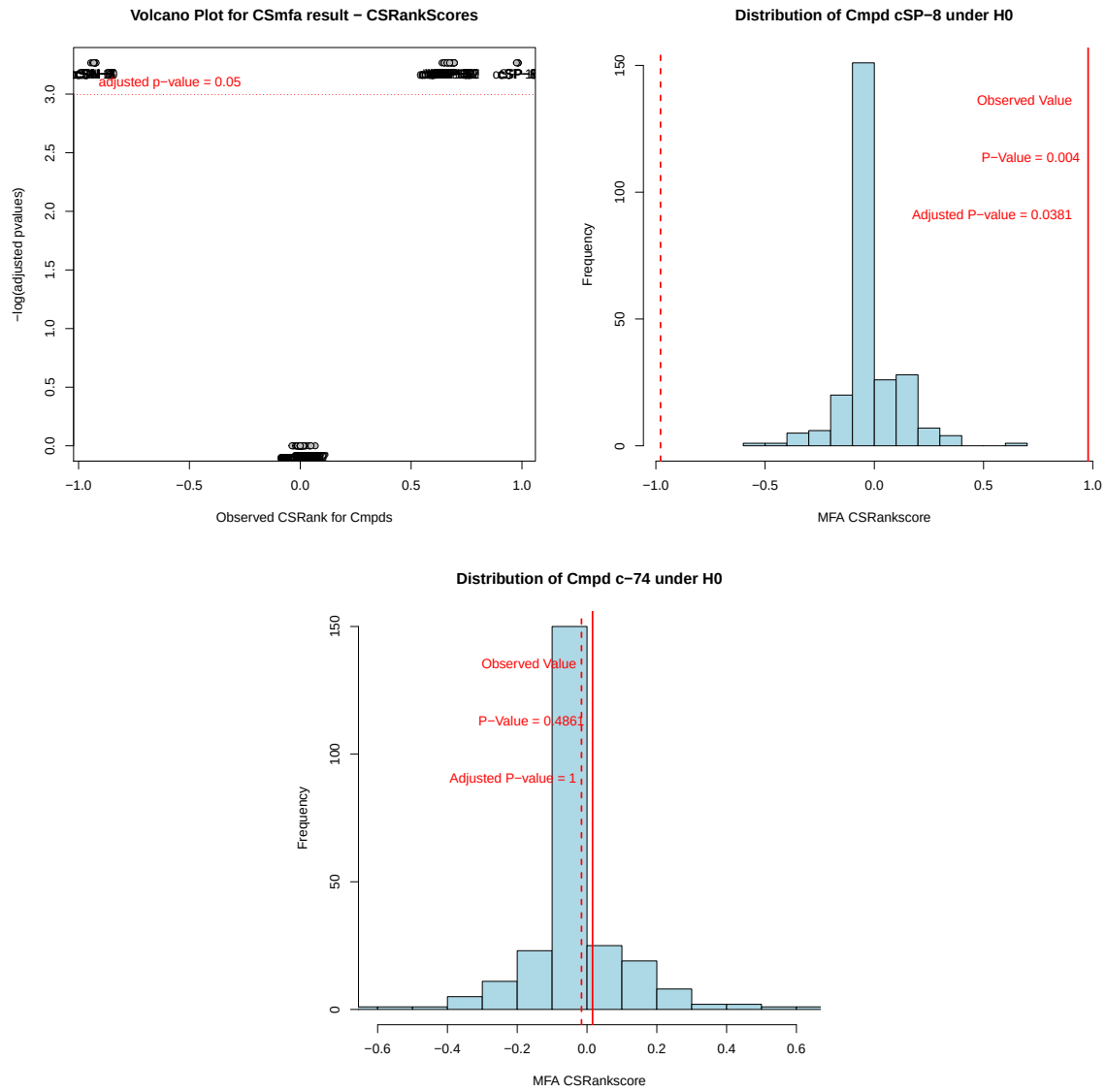

Figure 8: CSpermute graphs for MFA result

5 Example Compare CS Results

Finally, CSFA also provides a way to quickly compare the 2 results on the same data.
In the R-code below, we first compare the Zhang and Gant results with the MFA result. With `component2.plot=1` we choose the first component for the second result which is the first factor for the MFA result in this example. Since the Zhang and Gant analysis only provides connectivity scores, only 1 comparison graph will be created. The second example in the code compares the MFA with the FABIA results. For both results we choose the first component which corresponds with the first factor and first bicluster. Further, we also set some positive and negative gene thresholds for both of the results. In this example we keep them the same for both the MFA and FABIA results namely 2 for the upper threshold and -2 for the lower one. This time since both results also contain gene scores, 2 graphs will be created. Further because we set thresholds for the genes, the gene score comparison plot will be colored according to these thresholds.

```
comp_ZG_MFA <- CScompare(out_ZG,out_MFA,component2.plot=1,plot.type="sweave")

comp_MFA_FABIA <- CScompare(out_MFA,out_FABIA,component1.plot=1,component2.plot=1,
                gene.thresP=c(2,2),gene.thresN=c(-2,-2),plot.type="sweave")

comp_ZG_MFA[[1]]

## $scores
```

```
##                       CLoadings CRankScores GeneScores
## Correlation_Pearson  0.9965602   0.9982107         NA
## Correlation_Spearman 0.9627536   0.7273489         NA
##
## $pvalues
##                       CLoadings CRankScores
## Correlation_Pearson  0.9532248   0.9113716
## Correlation_Spearman 0.8737479   0.7128987
##
## $adj.pvalues
##                       CLoadings CRankScores
## Correlation_Pearson  0.9509475    0.946321
## Correlation_Spearman 0.5533918    0.542450


comp_MFA_FABIA[[1]]

## $scores
##                        CLoadings CRankScores GeneScores
## Correlation_Pearson  -0.9573119   0.9801271 -0.7752930
## Correlation_Spearman -0.5693159   0.5550448 -0.7996527
##
## $pvalues
## NULL
##
## $adj.pvalues
## NULL
```

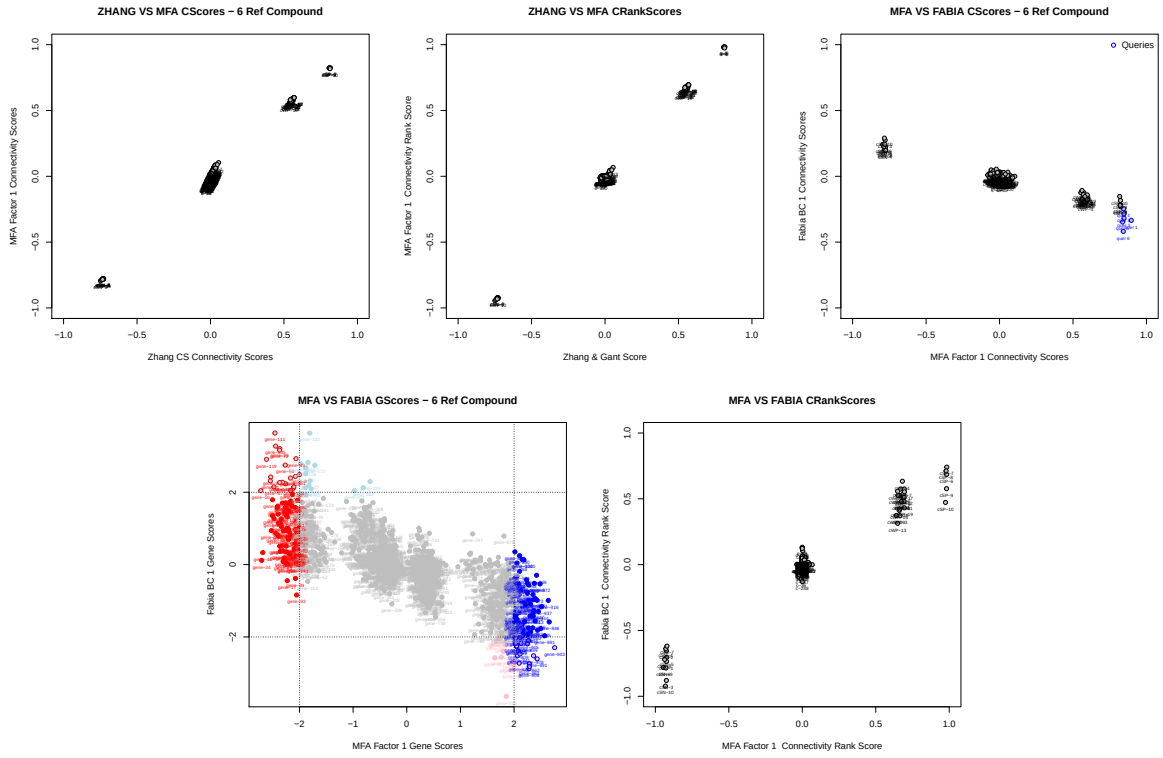

Figure 9: Compare CSresults

Note that apart from the scatter plots in Figure 9, the function also returns the pearson correlation between the (rank of the) scores.

Further because both the MFA and ZG contain p-values and adjusted p-values, the returned object also contains a small comparison between the number of significant p-values. The significancy threshold can be changed with the `threshold.pvalues` parameter and is defaulted to 0.05.

```
comp_ZG_MFA[[2]]

## $pvalues
##                 Result1.Sign Result1.NotSign
## Result2.Sign              35               0
## Result2.NotSign            9             291
##
## $adj.pvalues
##                 Result1.Sign Result1.NotSign
## Result2.Sign              35               0
## Result2.NotSign            1             299
```

References

Abdi, H., Williams, L. J., and Valentin, D. (2013), "Multiple factor analysis: principal component analysis for multitable and multiblock data sets," *WIREs Comput Stat*, 1–31.

Hochreiter, S., Bodenhoger, U., Heusel, M., Mayr, A., Mitterecker, A., Kasim, A., Khamiakova, T., Sanden, S., Lin, D., Talloen, W., Bijnens, Göhlmann, H., Shkedy, Z., and Clevert, D.-A. (2010), "FABIA: Factor Analysis for Bicluster acquisition," *Bioinformatics*, 26, 1520–1527.

Khamiakova, T. (2013), "Statistical Methods for Analysis of High Throughput Experiments in Early Drug Development," Ph.D. thesis, Hasselt University.

Lamb, J., Crawford, E. D., Peck, D., Modell, J. D., Blat, I. C., Wrobel, M. J., Lerner, J., Brunet, J.-P., Subramanian, A., Ross, K. N., Reich, M., Hieronymus, H., Wei, G., Armstrong, S. A., Haggarty, S. J., Clemons, P. A., Wei, R., Carr, A., Lander, E. S., and Golub, T. R. (2006), "The Connectivity Map: Using Gene-Expression Signatures to Connect Small Molecules, Genes, and Disease," *Science*, 313, 1929–1934.

Verbist, B., Klambauer, G., Vervoort, L., Talloen, W., The QSTAR Consortium, Shkedy, Z., Thas, O., Bender, A., Göhlmann, H., and Hochreiter, S. (2015), "Using transcriptomics to guide lead optimization in drug discovery projects: Lessons learned from the QSTAR project," *Drug Discovery Today*, 0, 1–9.

Zhang, S.-D. and Gant, T. W. (2008), "A simple and robust method for connecting small-molecule drugs using gene-expression signatures," *BMC Bioinformatics*, 9, 10.