Gene set analyses with the gCMAP package

Thomas Sandmann and Richard Bourgon

February 8, 2013

Contents

1 Introduction

The gCMAP package offers unified access to a number of different algorithms to compare a sets of genes across expression profiling experiments. It extends the functionality of the GSEABase package, which provides functions to generate and combine GeneSets from various sources.

2 CMAPCollection, SignedGeneSet and CMAPResults classes

Them gCMAPpackage introduces three new classes:

- SignedGeneSet: Extends the GeneSet class, with an additional geneSign slot to distinguish up- and downregulated set members.
- CMAPCollection: Is derived from the eSet class for efficient storage of large numbers of gene sets and related annotations.
- CMAPResults: Provides a unified output class for different gene set enrichment analysis methods.

2.1 CMAPCollections

To evaluate large gene sets collections containing thousands of gene sets, the gCMAP package introduces a new class CMAPCollections , to store gene sets and their relationships with each other in the form of a (sparse) incidence matrix. A derivative of the eSet class, a CMAPCollection also stores gene and gene set annotations in its featureData and phenoData slots.

CMAPCollections can be created de novo, e.g. with the newCMAPCollection function, or by coercing existing GeneSet, SignedGeneSet or GeneSetCollection objects. Often, large data matrices e.g. containing differential expression data from many different experiments, are available. The induceCMAPCollection function can be used to define gene sets from any eSet object by applying a user-defined threshold.

The gCMAPData NChannelSet object stores the results of three perturbation experiments, stimulation of tissue culture cells with drug1, drug2 or drug3. For each experiment, log2 fold change, z-scores and p-values (from differential expression analysis with the limma package) are available.

```
> library(gCMAP)
> data( gCMAPData ) ## example NChannelSet
> sampleNames( gCMAPData )

[1] "drug1" "drug2" "drug3"

> channelNames( gCMAPData )

[1] "log_fc" "p"      "z"
```

To induce gene sets of interest, a data slot and thresholds must be chosen.

```
> ## select all genes with z-scores > 2 or < -2
> cmap <- induceCMAPCollection( gCMAPData, element="z", lower=-2, higher=2)
> cmap
```

```
CMAPCollection (storageMode: lockedEnvironment)
assayData: 1000 features, 3 samples
  element names: members
protocolData: none
phenoData
  sampleNames: drug1 drug2 drug3
  varLabels: UID signed
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

```
> pData( cmap )
```

```
      UID signed
drug1   1   TRUE
drug2   2   TRUE
drug3   3   TRUE
```

The sign of the differential expression (e.g. the sign of the z-score or log2 fold change) is stored in the sparseMatrix stored as `assayData` in the `CMAPCollection` . Up-regulated gene set members are indicated by +1, down-regulated members by -1.

```
> head( members( cmap ) )
```

```
6 x 3 sparse Matrix of class "dgCMatrix"
       drug1 drug2 drug3
gene.1     .     .     .
gene.2     .     .     .
gene.3     .     .     .
gene.4    -1     .     .
gene.5     .     .     .
gene.6     .    -1     .
```

Sometimes, e.g. when selecting gene sets based on p-values, no sign information is available and all set members will simply be indicated with +1. To distinguish sets without sign information from those only containing up-regulated members, the `signed` column of the phenoData slot indicates how the information should be interpreted.

```
> signed( cmap )
```

```
drug1 drug2 drug3
 TRUE  TRUE  TRUE
```

As for other eSet-like objects, CMAPCollections can be subset to extract specific genes or gene sets.

```
> dim( cmap )
```

```
Features  Samples
    1000        3
```

```
> cmap[,1] ## the first gene set
```

```
CMAPCollection (storageMode: lockedEnvironment)
assayData: 1000 features, 1 samples
  element names: members
protocolData: none
phenoData
  sampleNames: drug1
  varLabels: UID signed
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

3 Comparing GeneSets

To compare the list of geneIds present in different CMAPCollections , GeneSets or GeneSetCollections , the Fisher test can be used. In addition to the GeneSets of interest, we also need to provide information about the gene 'universe', the complete ensemble of genes that could potentially be included in any set, e.g. all genes for which probes are available on a microarray, etc. Here, we will use all identifiers present in the gCMAPData dataset to define the gene identifier universe.

The following example compares the first gene set in our CMAPCollecttion to all three included sets. (In this vignette, we will refer to 'query' and 'target' objects. Every query object is compared individually to all targets and the results are returned in a single object.)

```
> universe <- featureNames( gCMAPData )
> results <- fisher_score(cmap[,1], cmap, universe)
> results
```

```
 CMAPResults object with the following data slots:
 set, trend, pval, padj, effect, nSet, nFound, UID, signed
 for 3 gene sets.
 1 test(s) obtained an adjusted p-value < 0.05

 Results from Fisher exact tests.
 P-values were adjusted using the 'p.adjust' function with method 'BH'.

    set trend          pval          padj    effect nSet nFound UID signed
1 drug1  over 2.132837e-210 6.398511e-210       Inf  190    190   1  FALSE
2 drug2  over  2.415470e-01  3.623206e-01 0.3328643   83     20   2  FALSE
3 drug3  over  6.880119e-01  6.880119e-01 0.1576603   42      9   3  FALSE
... (only top 5 results shown, use 'cmapTable' function to see all) ...
```

The fisher_score method returns a CMAPResults object, used by all analysis methods supported by the gCMAP package.

3.1 CMAPResults

Each CMAPResults object contains three elements

– An AnnotatedDataFrame called 'table', storing the results of comparing one query to all of the targets. Additional columns can be used to store information about the target gene sets. The supported gene set enrichment analysis methods return various scores, effect sizes and p-values, documented in the `varMetadata` slot of the 'table'. They can be accessed with the `labels` method.

- A 'docs' `character` vector to record information about the analysis run as a whole.
- A `list` 'errors', where potential warnings and error messages can be stored.

To `cmapTable` method returns the full result table, including annotation columns (if present) and labels. Individual accessors have been to return the p-value columns (`pval` or `padj`), effect size (`effect`) or to transform the adjusted p-values to z-scores on a standard normal scale (`zscores`).

```
> cmapTable( results )
```

```
    set trend          pval          padj    effect nSet nFound UID signed
1 drug1  over 2.132837e-210 6.398511e-210       Inf  190    190   1  FALSE
2 drug2  over  2.415470e-01  3.623206e-01 0.3328643   83     20   2  FALSE
3 drug3  over  6.880119e-01  6.880119e-01 0.1576603   42      9   3  FALSE
```

```
> labels( results )
```

```
                                 labelDescription
set                                       SetName
trend           Deviation from random expectation
pval                  Fisher's exact test p-value
padj                        Adjusted p-value (BH)
effect                                   Log-odds
nSet   Number of genes annotated in the query set
nFound  Number of query genes found in target set
UID                                           UID
signed                                     signed
```

```
> pval( results )
```

```
        drug1         drug2         drug3
2.132837e-210  2.415470e-01  6.880119e-01
```

```
> zscores( results )
```

```
     drug1      drug2      drug3
30.9201734  0.9109522  0.4015545
```

Several gene set enrichment analyses support many-to-many comparisons, including fisher_score. In this case, we receive a list of multiple `CMAPResults` objects, one for each element of the query. Each CMAPResults object contains the results for all query gene sets ordered by p-value. To extract individual slots from all CMAPResult objects in the list, e.g. with `sapply` , we must ensure that all results are returned in the same order, e.g. ordered by sampleNames.

```
> result.list <- fisher_score( cmap, cmap, universe )
> class( result.list )
```

```
[1] "list"
```

```
> length( result.list )
```

```
[1] 3
```

```
> class( result.list[[1]] )
```

```
[1] "CMAPResults"
attr(,"package")
[1] "gCMAP"
```

```
> ## all pairwise adjusted p-values
> sapply(result.list, function( x ) {
+        padj( x )[ sampleNames( cmap )]
+        })
```

```
              drug1         drug2        drug3
drug1 6.398511e-210  2.415470e-01 6.880119e-01
drug2  3.623206e-01 3.922263e-123 2.271885e-01
drug3  6.880119e-01  2.271885e-01 1.009276e-74
```

4 Differential expression analysis with gene sets

Frequently, we are interested in differential expression of gene sets across two or more conditions. The gCMAP package currently provides unified access to the sample-label permutation strategy implemented in the GSEAlm package, as well as multiple functions from the limma package: camera , romer and mroast . (For a detailed explanation oft the different methods, please consult the help entries of the original packages directly.)

For all methods, pre-processed expression data can be supplied as a data matrix, an ExpressionSet or any other eSet derivative. To perform a differential expression analysis, the experimental design must be specified, either by providing a design matrix directly or, for eSet or ExpressionSet objects, as a character string matching a phenoData column name.

Let's generate an matrix with random expression values, three treated and three control samples:

```
> ## random score matrix
> y <- matrix(rnorm(1000*6),1000,6,
+              dimnames=list( featureNames( gCMAPData ), 1:6 ))
> predictor <- c( rep("Control", 3), rep("Case", 3))
```

along with a CMAPCollection containg four unsigned gene sets, the first of which is actually differentially up-regulated in the 'Case' group.

```
> m <-replicate(4, {
+   s <- rep(0,1000)
+   s[ sample(1:1000, 20)] <-  1
+   s[ sample(1:1000, 20)] <- -1
+   s
+   })
> dimnames(m) <- list(row.names( y ),
+                     paste("set", 1:4, sep=""))
> ## Set1 is up-regulated
> y[,c(4:6)] <- y[,c(4:6)] + m[,1]*2
> ## create CMAPCollection
> cmap <- CMAPCollection(m, signed=rep(TRUE,4))
```

The gCMAP package offers four different algorithms to test for differential expression between the 'control' and 'treatment' samples:

```
> gsealm_score(y, cmap, predictor=predictor, nPerm=100)
```

```
 CMAPResults object with the following data slots:
 set, trend, pval, padj, effect, nSet, nFound, geneScores, signed
 for 4 gene sets.
 0 test(s) obtained an adjusted p-value < 0.05
 GSEAlm analysis with formula ~predictor using 100 sample label permutations.
 P-values were adjusted with the 'p-adjust' function using method 'BH'.

   set          trend       pval       padj      effect nSet nFound signed
1 set1 anticorrelated 0.03960396 0.07920792 -18.5941433   40     40   TRUE
2 set4     correlated 0.03960396 0.07920792   1.7413761   39     39   TRUE
3 set3     correlated 0.08910891 0.11881188   1.9710559   40     40   TRUE
4 set2 anticorrelated 0.43564356 0.43564356  -0.1898484   39     39   TRUE
... (only top 5 results shown, use 'cmapTable' function to see all) ...
```

```
> mroast_score(y, cmap, predictor=predictor)
```

```
 CMAPResults object with the following data slots:
 set, trend, pval, padj, nSet, geneScores, signed
 for 12 gene sets.
 2 test(s) obtained an adjusted p-value < 0.05

 All results, including adjusted p-values, were obtained
 with the 'mroast' function from the 'limma' package..

   set trend  pval      padj nSet signed
1 set1 Mixed 0.001 0.0020000   40  FALSE
2 set1    Up 0.009 0.0340000   40  FALSE
3 set3 Mixed 0.112 0.2230000   40  FALSE
4 set2 Mixed 0.216 0.2873333   39  FALSE
5 set4  Down 0.241 0.9340000   39  FALSE
... (only top 5 results shown, use 'cmapTable' function to see all) ...
```

Both gsealm_score and mroast perform self-contained test. (Goeman and Buhlmann, 2007). (Please note that we only run 100 `gsealm_score` permutations to obtain a p-value in this example - in a real analysis, increasing this number, e.g. to 1000, is recommended.) In case a competitive hypothesis needs to be tested, the camera_score and romer_score methods (calling the `romer` and `camera` functions from the limma package, respectively) can be used instead.

```
> camera_score(y, cmap, predictor=predictor)
```

```
 CMAPResults object with the following data slots:
 set, trend, pval, padj, nSet, nFound, geneScores, signed
 for 12 gene sets.
 0 test(s) obtained an adjusted p-value < 0.05

 Results were obtained with the 'camera' function from the 'limma' package.
 P-values were adjusted with the 'p-adjust' function using method 'BH'.

   set    trend        pval       padj nSet nFound signed
1 set1       Up 0.006404952 0.07685943   40     40  FALSE
2 set1 TwoSided 0.012809905 0.07685943   40     40  FALSE
3 set4     Down 0.226795214 0.90718086   39     39  FALSE
4 set3       Up 0.315492068 0.90718086   40     40  FALSE
5 set2       Up 0.432675400 0.90718086   39     39  FALSE
... (only top 5 results shown, use 'cmapTable' function to see all) ...
```

```
> romer_score(y, cmap, predictor=predictor)
```

```
 CMAPResults object with the following data slots:
 set, trend, pval, padj, nSet, nFound, geneScores, signed
 for 12 gene sets.
 1 test(s) obtained an adjusted p-value < 0.05
 nResults obtained with the 'romer' function from the 'limma' package.
 P-values were adjusted with the 'p-adjust' function using method 'BH'.

   set trend   pval    padj nSet nFound signed
1 set1 Mixed 0.0001 0.00120   40     40  FALSE
2 set4  Down 0.1912 0.79884   39     39  FALSE
3 set1    Up 0.2193 0.79884   40     40  FALSE
4 set3    Up 0.3770 0.79884   40     40  FALSE
5 set3 Mixed 0.3942 0.79884   40     40  FALSE
... (only top 5 results shown, use 'cmapTable' function to see all) ...
```

Currently, only gsealm_score takes the sign of the gene set members (indicating whether a gene had originally be identified as up- or down-regulated) into account.

5 Analysis of individual score profiles

In addition to analyzing complete experiments, other approaches to gene set enrichment testing evaluate whether a given statistic for the members of a gene set ranked highly relative to random sets.

The wilcox_score method calculates the Wilcox-rank sum statistic, assessing whether the ranked scores of a gene set are enriched at the top or bottom of the complete list of scores.

The gsealm_jg_score calculates the mean score for all gene set members and provides a p-value based on the standard normal distribution (Jiang and Gentleman, 2007).

The connectivity_score is calculated according to Lamb, J. et al. (2006) and corresponds to the scaled score described in this publication. (It does not provide a p-value.)

For illustration, we compare the first column of z-scores stored in the gCMAPData `NChannelSet` to the three gene sets induced from the same dataset in the first section of this vignette..

```
> profile <- assayDataElement(gCMAPData[,1], "z") ## extract first column
> head(profile)
```

```
            drug1
gene.1 -0.4600253
gene.2 -1.8756099
gene.3 -0.7766186
gene.4 -2.9651795
gene.5 -1.2265235
gene.6 -0.1037107
```

```
> sampleNames(cmap) ## three gene sets
```

```
[1] "set1" "set2" "set3" "set4"
```

```
> gsealm_jg_score(profile, cmap)
```

```
 CMAPResults object with the following data slots:
 set, trend, pval, padj, effect, nSet, nFound, geneScores, signed
 for 4 gene sets.
 0 test(s) obtained an adjusted p-value < 0.05

 Parametric 'JG' score summary.
 P-values were adjusted with the 'p-adjust' function using method 'BH'.
```

```
   set          trend       pval       padj     effect nSet nFound signed
1 set3 anticorrelated 0.01980255 0.07921019 -2.3300682   40     40   TRUE
2 set1 anticorrelated 0.13605677 0.27211354 -1.4906372   40     40   TRUE
3 set4     correlated 0.60358170 0.69501998  0.5192568   39     39   TRUE
4 set2     correlated 0.69501998 0.69501998  0.3920517   39     39   TRUE
... (only top 5 results shown, use 'cmapTable' function to see all) ...
```

As expected the first gene set, which was derived from the same experiment as the profile, receives highly significant p-values.

Alternatively, the Wilcox Rank sum test or the original Connectivity Score can be calculated. (Please note that the `connectivity_score` does not return a p-value and is hard to interpret for a single profile.)

```
> wilcox_score(profile, cmap)

 CMAPResults object with the following data slots:
 set, trend, pval, padj, effect, nSet, nFound, geneScores, signed
 for 4 gene sets.
 0 test(s) obtained an adjusted p-value < 0.05

 Results from a two-tailed Wilcox-Rank Sum test
 p-values were adjusted using the 'p.adjust' function with method 'BH'.

   set          trend      pval      padj     effect nSet nFound signed
1 set3 anticorrelated 0.1063337 0.2258709 -1.2462644   40     40   TRUE
2 set1 anticorrelated 0.1129354 0.2258709 -1.2110640   40     40   TRUE
3 set4     correlated 0.2793444 0.3618659  0.5847903   39     39   TRUE
4 set2     correlated 0.3618659 0.3618659  0.3534757   39     39   TRUE
... (only top 5 results shown, use 'cmapTable' function to see all) ...
```

```
> connectivity_score(profile, cmap)

 CMAPResults object with the following data slots:
 set, trend, effect, nSet, nFound, geneScores, signed
 for 4 gene sets.
 Scores were calculated and scaled according to Lamb, J. et al. (2006).

   set trend   effect nSet nFound signed
1 set1  down -1.00000   40     40   TRUE
2 set4    up  1.00000   39     39   TRUE
3 set3  down -0.97733   40     40   TRUE
4 set2    up  0.00000   39     39   TRUE
... (only top 5 results shown, use 'cmapTable' function to see all) ...
```

6 Overview plots

When comparing a set of genes, a profile or a complete experiment to a large gene set collection, e.g. induced from the original Connectivity map data generated at the Broad institute (Lamb et al, Science, 2006), high level diagnostic plots can provide a first overview of the results.

For illustration purposes, we generate a random profile of z-scores for 1000 genes as well as CMAPCollection with a random set of 1000 gene sets. One of them, set1, is actually differentially regulated.

```
> ## create random score profile
> set.seed(123)
> z <- rnorm(1000)
> names(z) <- paste("g", 1:1000, sep="")
> ## generate random incidence matrix of gene sets
> n <-replicate(1000, {
+   s <- rep(0,1000)
+   s[ sample(1:1000, 20)] <-  1
+   s[ sample(1:1000, 20)] <- -1
+   s
+   })
> dimnames(n) <- list(names(z), paste("set",
+                                     1:1000,
+                                     sep=""))
> ## Set1 is up-regulated
> z <- z + n[,1]*2
> ## create CMAPCollection
> cmap.2 <- CMAPCollection(n, signed=rep(TRUE,1000))
```

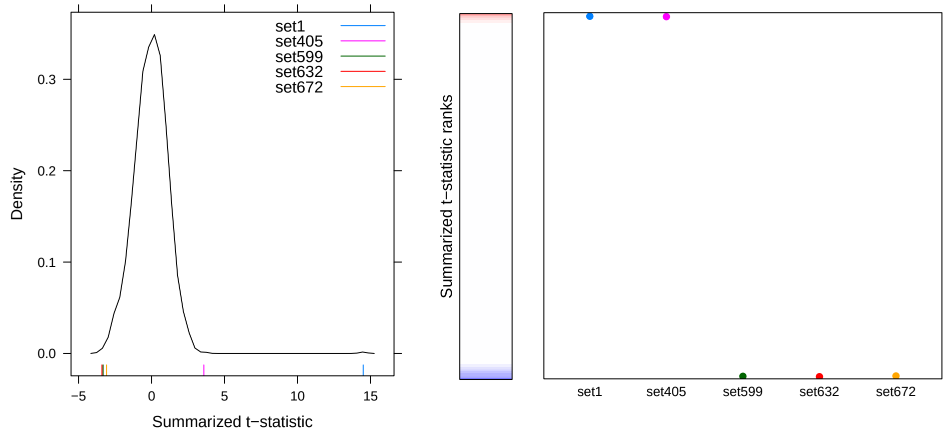
```
> ## gene-set enrichment test
> res <- gsealm_jg_score(z, cmap.2)
> class(res)
```

```
[1] "CMAPResults"
attr(,"package")
[1] "gCMAP"
```

```
> res
```

```
 CMAPResults object with the following data slots:
 set, trend, pval, padj, effect, nSet, nFound, geneScores, signed
 for 1000 gene sets.
 1 test(s) obtained an adjusted p-value < 0.05

 Parametric 'JG' score summary.
 P-values were adjusted with the 'p-adjust' function using method 'BH'.

     set          trend         pval         padj    effect nSet nFound signed
1   set1     correlated 1.534819e-47 1.534819e-44 14.483753   40     40   TRUE
2 set405     correlated 3.470647e-04 1.735323e-01  3.577373   40     40   TRUE
3 set632 anticorrelated 6.665787e-04 2.221929e-01 -3.402969   39     39   TRUE
4 set599 anticorrelated 8.892471e-04 2.223118e-01 -3.323408   40     40   TRUE
5 set672 anticorrelated 2.063107e-03 3.723693e-01 -3.080994   40     40   TRUE
... (only top 5 results shown, use 'cmapTable' function to see all) ...
```

```
> plot(res, strip.subset=1:5, pch=19)
```

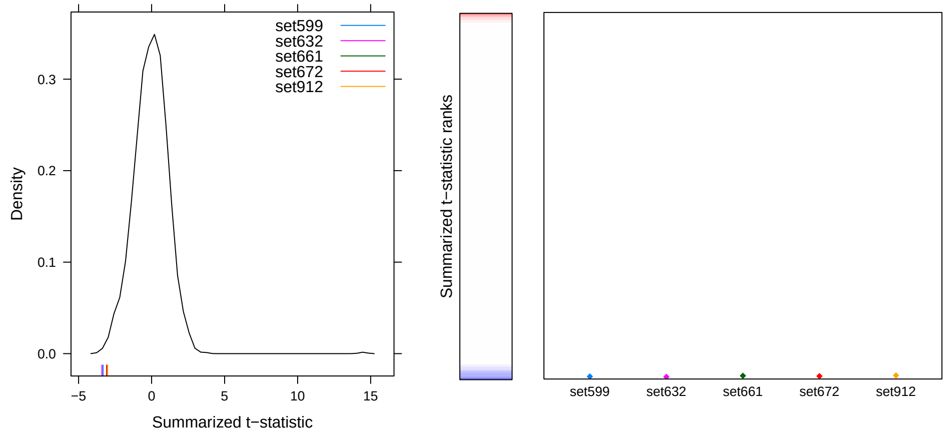


A call to the plot method on a CMAPResults object yields three graphical overviews: on the left, a density of all 1000 reported effect sizes, in this case JG-scores, is shown. In the absence of correlation between genes, this distribution follows a normal distribution. (While this is true for this set of randomly generated scores, the distribution of JG scores observed in practice is actually broader than expected, testament to the non-random patterns of gene expression.) As expected, set1 is the only set reported with an adjusted p-value of less than 0.05.

In the centre, a heatmap of the rank-ordered scores is displayed, with low and high scores displayed as blue and red stripes, respectively. By default, scores between -2 and 2 are shown in grey. To display scores above 2 or below -2, a color gradient from white to red or from white to blue is applied, respectively. (Both the choice of colors and thresholds of the color gradients can be configured, please see the CMAPResults help page for details.)

On the right, the ranks of the top five gene sets are highlighted. (The same set is also indicatd in the rug of the density distribution on the left.) The strip.subset parameter can be used to focus on specific sets of interest, e.g. by matching keywords in their annotation columns or by applying a threshold to any of the score columns. For example, we can highlight those sets with JG-scores of less than -3.

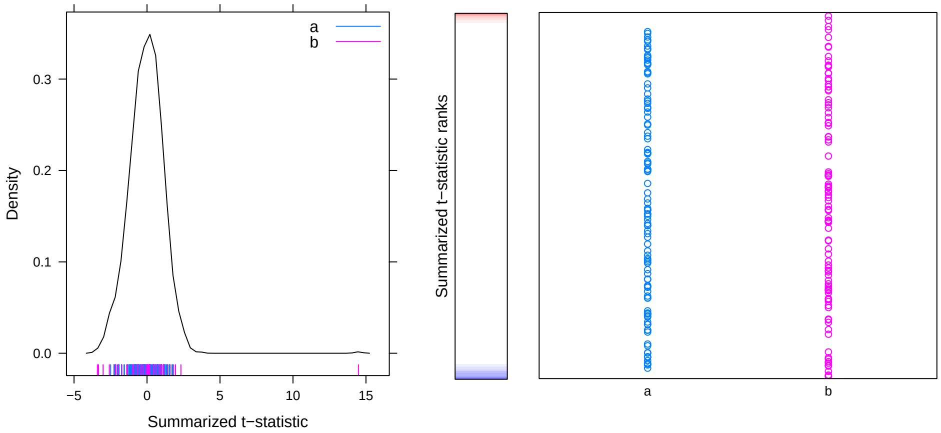
```
> sets.down <- effect( res ) < -3
> plot(res, strip.subset=sets.down, pch=18)
```



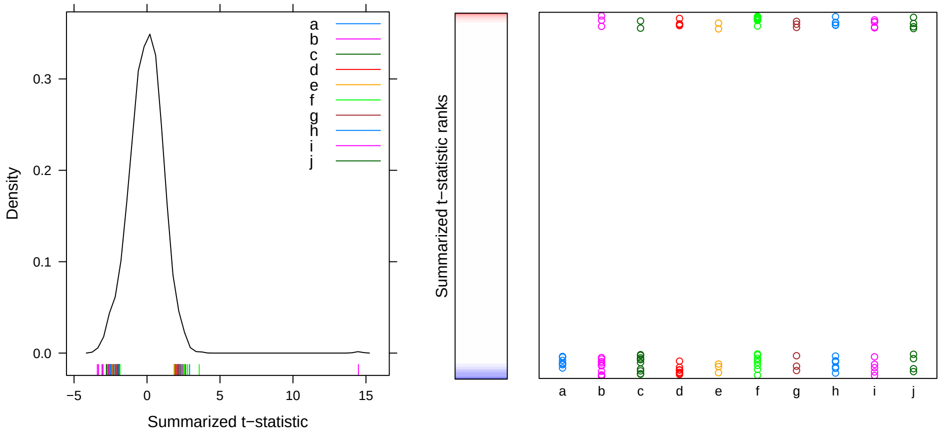
In addition, we can group by shared annotation entries by specifying a column of the CMAPResults object with the 'strip.anno' parameter. To demonstrate, we add a column of 10 class annotations to the 'res' CMAPResults object. Now, we can for example highlight the distribution of scores from two particular classes (e.g. 'a' and 'b'), or investigate the class membership of the top 100 sets.

(Please note that any annotation columns present in a queried CMAPCollection will be included in the CMAPResults object automatically.)

```
> res$class <- sample(letters[1:10], 1000, replace=TRUE)
> plot(res, strip.anno="class", strip.subset=which(res$class %in% c("a", "b")))
```



```
> plot(res, strip.anno="class", strip.subset=1:100)
```



7 Retrieving gene-level information

Once significantly enriched gene sets have been identified, we may want to take a closer look at the behavior of individual genes. Are expression changes associated with many gene set members or do specific genes respond particularly strongly ?

All methods implemented in the `gCMAP` package, with the exception of `fisher_score` , return gene-level scores when the optional 'keep.scores' parameter is set to 'TRUE'. To demonstrate, we repeat the `gsealm_score` call from above.

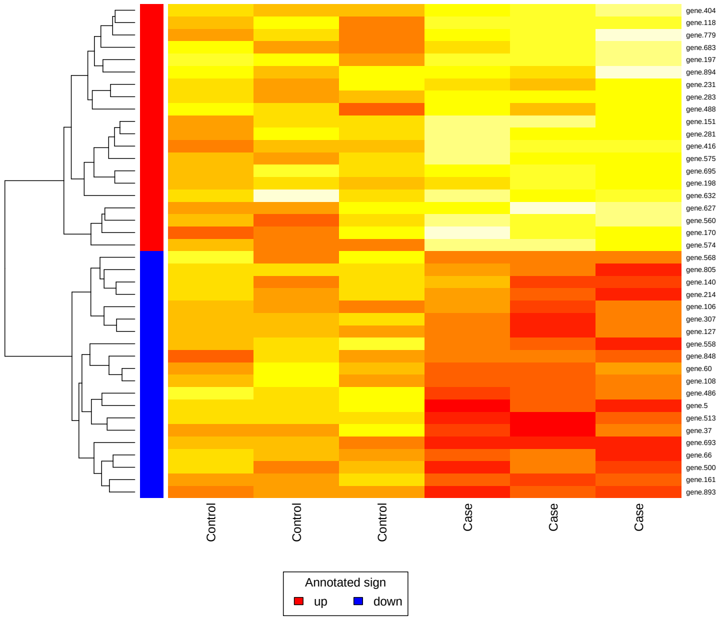
```
> res <- gsealm_score(y, cmap, predictor=predictor, nPerm=100, keep.scores=TRUE)
> res
 CMAPResults object with the following data slots:
 set, trend, pval, padj, effect, nSet, nFound, geneScores, signed
 for 4 gene sets.
 0 test(s) obtained an adjusted p-value < 0.05

 GSEAlm analysis with formula ~predictor using 100 sample label permutations.
 P-values were adjusted with the 'p-adjust' function using method 'BH'.

   set          trend       pval       padj      effect nSet nFound signed
1 set1 anticorrelated 0.02970297 0.05940594 -18.5941433   40     40   TRUE
2 set4     correlated 0.02970297 0.05940594   1.7413761   39     39   TRUE
3 set3     correlated 0.08910891 0.11881188   1.9710559   40     40   TRUE
4 set2 anticorrelated 0.41584158 0.41584158  -0.1898484   39     39   TRUE
     geneScores
1 40 x 6 matrix
2 39 x 6 matrix
3 40 x 6 matrix
4 39 x 6 matrix
... (only top 5 results shown, use 'cmapTable' function to see all) ...
```

```
> set1.expr <- geneScores(res)[["set1"]]
> head(set1.expr)
                   1            2          3          4         5          6
gene.5    0.55569105  0.484890061  1.1029639 -4.2226726 -1.690514 -3.0980189
gene.37  -0.30577334 -0.816284729  1.2852544 -2.8099300 -3.614972 -0.9834118
gene.60  -0.56709908  1.486837939  0.2300383 -2.2423362 -1.837553 -0.8539422
gene.66   0.50885474 -0.006704736 -0.7715582 -2.1887439 -1.281992 -3.4586807
gene.106  0.06942784 -0.921643964 -1.2380811 -0.3791616 -2.821877 -1.4603829
gene.108 -0.03259198  1.315295427 -0.4736140 -2.1384493 -1.940030 -1.3709854
```

Expression scores for each gene set are now available in the geneScores `cmapResults` colum, which can be accessed through a method with the same name. Each matrix of expression scores is accompanied by an additional 'sign' attribute to remind us whether gene set members were annotated as up- or down-regulated.

For example, we can now visualize the expression scores of set1 member genes in a heatmap. As expected, genes annotated as 'up-regulated' (red sidebar) show higher expression in Cases than Controls and the reverse is true for genes annotated as 'down-regulated' (blue sidebar).

```
> heatmap(set1.expr, scale="none", Colv=NA, labCol=predictor,
+         RowSideColors=ifelse( attr(set1.expr, "sign") == "up", "red", "blue"),
+         margin=c(7,5))
> legend(0.35,0,legend=c("up", "down"),
+        fill=c("red", "blue"),
+        title="Annotated sign", horiz=TRUE, xpd=TRUE)
```



Each row in the `CMAPResults` objects features an subset of the original query ExpressionSet. As genes can be part of many different genes sets, querying large gene set collections may result in storing duplicate data rows over and over again, considerably increasing the memory footprint of the `CMAPResults` object.

Alternatively, we can extract the scores from the original data source. For example, we can obtain a nested list of scores for all sets and data columns by passing the `CMAPCollection` (cmap) and the score matrix (y) to the `featureScores` method. The element for 'set1' corresponds to the score matrix we obtained above.

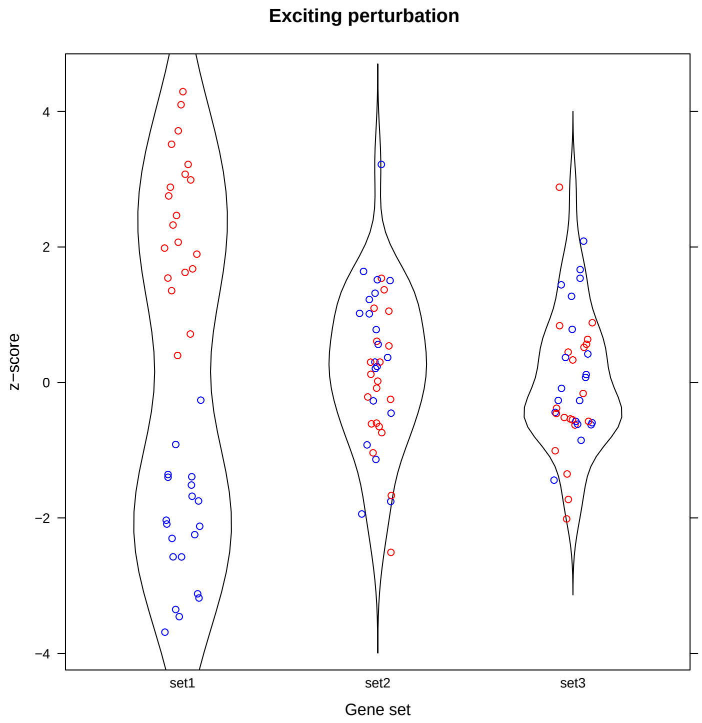
```
> res <- featureScores(cmap, y)
> class(res)

[1] "list"

> names(res)

[1] "set1" "set2" "set3" "set4"

> identical( res[["set1"]], set1.expr )

[1] TRUE
```

Different scores for the same gene set can be handled conveniently as matrices, because all score vectors are of the same length. In a different scenario, we might want to compare expression changes for different gene sets, likely containing different numbers of genes. Reversing the order of the arguments passed to the `featureScores` function returns a list of score vectors for each perturbation.

The `melt` function from the `reshape` package offers a simple way to convert it into a data.frame as input e.g. for plot functions from the `lattice` or `ggplot2` package. (For more on handling of nested lists, you may want to take a look at the `plyr` package.)

To demonstrate, we return to the z-score vector 'z' and the associated `CMAPCollection` 'cmap.2' we generated in the section 'Overview plots'.

```
> ## retrieve scores for the first three sets
> res <- featureScores(z, cmap.2[,1:3]) ## reversed argument order
> res <- res[[1]]
> ## transform into data.frame
> require( reshape )
> df <- melt( res )
> ## collect gene signs
> signs <- as.vector(sapply(res, attr, "sign"))
> require( lattice )
> bwplot(value ~ L1, data=df,
+     xlab="Gene set", ylab="z-score",
+     main="Exciting perturbation",
+     panel = function(..., box.ratio) {
+     panel.violin(..., col = "transparent",
+     varwidth = FALSE, box.ratio = box.ratio)
+     panel.xyplot(..., jitter.x=TRUE, fill = NULL,
+     col=ifelse( signs == "up", "red", "blue"))
+     })
```



```
>   sessionInfo()

R version 2.15.2 (2012-10-26)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=C                 LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] reshape_0.8.4        plyr_1.8             gCMAP_1.1.7
 [4] DESeq_1.10.1         lattice_0.20-13      locfit_1.5-8
 [7] GSEABase_1.20.2      graph_1.36.2         annotate_1.36.0
[10] AnnotationDbi_1.20.3 Biobase_2.18.0       BiocGenerics_0.4.0

loaded via a namespace (and not attached):
 [1] DBI_0.2-5             GSEAlm_1.18.0         IRanges_1.16.4
 [4] Matrix_1.0-10         RColorBrewer_1.0-5    RSQLite_0.11.2
 [7] XML_3.95-0.1          bigmemory_4.3.0       bigmemory.sri_0.1.2
[10] bigmemoryExtras_1.0.0 genefilter_1.40.0     geneplotter_1.36.0
[13] grid_2.15.2           latticeExtra_0.6-24   limma_3.14.4
[16] parallel_2.15.2       splines_2.15.2        stats4_2.15.2
[19] survival_2.37-2       tools_2.15.2          xtable_1.7-0
```