Gene set analyses with the gCMAP package

Thomas Sandmann and Richard Bourgon

October 17, 2016

This document exemplifies the structure and use of the classes and (some of) the methods offered by the gCMAP package. For information on performing queries via an R-only, custom web application and to access a tutorial with examples using real biological datasets available from public databases, please refer to the documentation of the gCMAPWeb companion package.

Contents

1 Introduction 2
2 CMAPCollection, SignedGeneSet and CMAPResults classes 2
   2.1 CMAPCollections 2
3 Comparing GeneSets 4
   3.1 CMAPResults 4
4 Differential expression analysis with gene sets 6
5 Analysis of individual score profiles 8
6 Overview plots 10
7 Retrieving gene-level information 11
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

The gCMAP package 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.

```r
> 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.

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

2
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
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 CMAPCollection 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.)

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

CMAPResults object with the following data slots:
- set, trend, pval, padj, effect, LOR, nSet, nFound, geneScores, 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'.

<table>
<thead>
<tr>
<th>set</th>
<th>trend</th>
<th>pval</th>
<th>padj</th>
<th>effect</th>
<th>LOR</th>
<th>nSet</th>
<th>nFound</th>
<th>UID</th>
</tr>
</thead>
<tbody>
<tr>
<td>drug1</td>
<td>over</td>
<td>2.132837e-210</td>
<td>6.398511e-210</td>
<td>30.96</td>
<td>Inf</td>
<td>190</td>
<td>190</td>
<td>1</td>
</tr>
<tr>
<td>drug2</td>
<td>over</td>
<td>2.415470e-01</td>
<td>3.623206e-01</td>
<td>1.17</td>
<td>0.3328643</td>
<td>83</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>drug3</td>
<td>over</td>
<td>6.880119e-01</td>
<td>6.880119e-01</td>
<td>0.40</td>
<td>0.1576603</td>
<td>42</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

signed
1 FALSE
2 FALSE
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 ).

```r
> cmapTable( results )
set trend pval padj effect LOR nSet nFound UID
1 drug1 over 2.132837e-210 6.398511e-210 30.96 Inf 190 190 1
2 drug2 over 2.415470e-01 3.623206e-01 1.17 0.3328643 83 20 2
3 drug3 over 6.880119e-01 6.880119e-01 0.40 0.1576603 42 9 3
signed
1 FALSE
2 FALSE
3 FALSE
```

```r
> labels( results )
labelDescription
set SetName
trend Deviation from random expectation
pval Fisher's exact test p-value
padj Adjusted p-value (BH)
effect z-score based on the standard normal distribution
LOR Log Odds Ratio
nSet Number of genes annotated in the reference set
nFound Number of genes found in query and target sets
geneScores Identifiers of genes found in query and target sets
UID UID
signed signed
```

```r
> pval( results )
drug1 drug2 drug3
2.132837e-210 2.415470e-01 6.880119e-01
```

```r
> 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.

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

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

```r
> 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'.

<table>
<thead>
<tr>
<th>set</th>
<th>trend</th>
<th>pval</th>
<th>padj</th>
<th>effect</th>
<th>nSet</th>
<th>nFound</th>
<th>signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>set1</td>
<td>anticorrelated</td>
<td>0.03960396</td>
<td>0.07920792</td>
<td>-18.594133</td>
<td>40</td>
<td>40</td>
<td>TRUE</td>
</tr>
<tr>
<td>set2</td>
<td>correlated</td>
<td>0.03960396</td>
<td>0.07920792</td>
<td>1.7435310</td>
<td>39</td>
<td>39</td>
<td>TRUE</td>
</tr>
<tr>
<td>set3</td>
<td>correlated</td>
<td>0.08910891</td>
<td>0.11881188</td>
<td>1.9710559</td>
<td>40</td>
<td>40</td>
<td>TRUE</td>
</tr>
<tr>
<td>set4</td>
<td>anticorrelated</td>
<td>0.43564356</td>
<td>0.43564356</td>
<td>-0.1876935</td>
<td>39</td>
<td>39</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

... (only top 5 results shown, use `cmapTable` function to see all) ...

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

CMAPResults object with the following data slots:
- set
- trend
- pval
- padj
- nSet
- geneScores
- signed

for 4 gene sets.
0 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..

<table>
<thead>
<tr>
<th>set</th>
<th>trend</th>
<th>pval</th>
<th>padj</th>
<th>nSet</th>
<th>signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>set1</td>
<td>Up</td>
<td>0.017</td>
<td>0.0660</td>
<td>40</td>
<td>FALSE</td>
</tr>
<tr>
<td>set2</td>
<td>Down</td>
<td>0.506</td>
<td>0.7745</td>
<td>39</td>
<td>FALSE</td>
</tr>
<tr>
<td>set3</td>
<td>Up</td>
<td>0.659</td>
<td>0.7745</td>
<td>40</td>
<td>FALSE</td>
</tr>
<tr>
<td>set4</td>
<td>Up</td>
<td>0.775</td>
<td>0.7750</td>
<td>39</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

... (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.

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

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 were obtained with the `camera` function from the `limma` package..

<table>
<thead>
<tr>
<th>set</th>
<th>trend</th>
<th>pval</th>
<th>padj</th>
<th>effect</th>
<th>nSet</th>
<th>nFound</th>
<th>signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>set1</td>
<td>Up</td>
<td>0.1337796</td>
<td>0.5351185</td>
<td>-0.021582596</td>
<td>40</td>
<td>40</td>
<td>FALSE</td>
</tr>
<tr>
<td>set2</td>
<td>Up</td>
<td>0.6309841</td>
<td>0.8441086</td>
<td>0.004394891</td>
<td>40</td>
<td>40</td>
<td>FALSE</td>
</tr>
<tr>
<td>set3</td>
<td>Down</td>
<td>0.6330815</td>
<td>0.8441086</td>
<td>-0.016781422</td>
<td>39</td>
<td>39</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
> 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'.

<table>
<thead>
<tr>
<th>set</th>
<th>trend</th>
<th>pval</th>
<th>padj</th>
<th>nSet</th>
<th>nFound</th>
<th>signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>set1</td>
<td>Mixed</td>
<td>0.0001</td>
<td>0.00120</td>
<td>40</td>
<td>40</td>
<td>FALSE</td>
</tr>
<tr>
<td>set4</td>
<td>Down</td>
<td>0.1899</td>
<td>0.78680</td>
<td>39</td>
<td>39</td>
<td>FALSE</td>
</tr>
<tr>
<td>set1</td>
<td>Up</td>
<td>0.1967</td>
<td>0.78680</td>
<td>40</td>
<td>40</td>
<td>FALSE</td>
</tr>
<tr>
<td>set3</td>
<td>Up</td>
<td>0.3776</td>
<td>0.79488</td>
<td>40</td>
<td>40</td>
<td>FALSE</td>
</tr>
<tr>
<td>set3</td>
<td>Mixed</td>
<td>0.3887</td>
<td>0.79488</td>
<td>40</td>
<td>40</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

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 Wilcoxon-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'.

<table>
<thead>
<tr>
<th>set</th>
<th>trend</th>
<th>pval</th>
<th>padj</th>
<th>effect</th>
<th>nSet</th>
<th>nFound</th>
<th>signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>set3</td>
<td>anticorrelated</td>
<td>0.01980255</td>
<td>0.07921019</td>
<td>-2.3300682</td>
<td>40</td>
<td>40</td>
<td>TRUE</td>
</tr>
<tr>
<td>set1</td>
<td>anticorrelated</td>
<td>0.13605677</td>
<td>0.27211354</td>
<td>-1.4906372</td>
<td>40</td>
<td>40</td>
<td>TRUE</td>
</tr>
<tr>
<td>set4</td>
<td>correlated</td>
<td>0.59196290</td>
<td>0.68269474</td>
<td>0.5359937</td>
<td>39</td>
<td>39</td>
<td>TRUE</td>
</tr>
<tr>
<td>set2</td>
<td>correlated</td>
<td>0.68269474</td>
<td>0.68269474</td>
<td>0.4087887</td>
<td>39</td>
<td>39</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

... (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'.

<table>
<thead>
<tr>
<th>set</th>
<th>trend</th>
<th>pval</th>
<th>padj</th>
<th>effect</th>
<th>nSet</th>
<th>nFound</th>
<th>signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>set3</td>
<td>anticorrelated</td>
<td>0.1063337</td>
<td>0.2258709</td>
<td>-1.2462644</td>
<td>40</td>
<td>40</td>
<td>TRUE</td>
</tr>
<tr>
<td>set1</td>
<td>anticorrelated</td>
<td>0.1129354</td>
<td>0.2258709</td>
<td>-1.2110640</td>
<td>40</td>
<td>40</td>
<td>TRUE</td>
</tr>
<tr>
<td>set4</td>
<td>correlated</td>
<td>0.2793444</td>
<td>0.3618659</td>
<td>0.5847903</td>
<td>39</td>
<td>39</td>
<td>TRUE</td>
</tr>
<tr>
<td>set2</td>
<td>correlated</td>
<td>0.3618659</td>
<td>0.3618659</td>
<td>0.3534757</td>
<td>39</td>
<td>39</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

... (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).

<table>
<thead>
<tr>
<th>set</th>
<th>trend</th>
<th>nSet</th>
<th>nFound</th>
<th>signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>set1</td>
<td>down</td>
<td>40</td>
<td>40</td>
<td>TRUE</td>
</tr>
<tr>
<td>set4</td>
<td>up</td>
<td>39</td>
<td>39</td>
<td>TRUE</td>
</tr>
<tr>
<td>set3</td>
<td>down</td>
<td>40</td>
<td>40</td>
<td>TRUE</td>
</tr>
<tr>
<td>set2</td>
<td>up</td>
<td>39</td>
<td>39</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

... (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.

```r
> ## 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'.

<table>
<thead>
<tr>
<th>set</th>
<th>trend</th>
<th>pval</th>
<th>padj</th>
<th>effect</th>
<th>nSet</th>
<th>nFound</th>
<th>signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>set1</td>
<td>correlated</td>
<td>1.534819e-47</td>
<td>1.534819e-44</td>
<td>14.483753</td>
<td>40</td>
<td>40</td>
<td>TRUE</td>
</tr>
<tr>
<td>set405</td>
<td>correlated</td>
<td>3.470647e-04</td>
<td>1.735323e-01</td>
<td>3.577373</td>
<td>40</td>
<td>40</td>
<td>TRUE</td>
</tr>
<tr>
<td>set632</td>
<td>anticorrelated</td>
<td>6.603058e-14</td>
<td>3.405551</td>
<td>-3.405551</td>
<td>39</td>
<td>39</td>
<td>TRUE</td>
</tr>
<tr>
<td>set599</td>
<td>anticorrelated</td>
<td>8.892471e-04</td>
<td>2.233118e-01</td>
<td>-3.323408</td>
<td>40</td>
<td>40</td>
<td>TRUE</td>
</tr>
<tr>
<td>set672</td>
<td>anticorrelated</td>
<td>2.063107e-03</td>
<td>3.723693e-01</td>
<td>-3.080994</td>
<td>40</td>
<td>40</td>
<td>TRUE</td>
</tr>
</tbody>
</table>
```

(only top 5 results shown, use 'cmapTable' function to see all) ...
A call to the `plot` method on a `CMAPResults` object yields two 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.)

On the right, 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 -3 and 3 are hidden. To display scores above 3 or below -3, 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.)

```r
> sets.down <- effect(res) < -3
> plot(res)
```

### 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.

```r
> 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'.

<table>
<thead>
<tr>
<th>set</th>
<th>trend</th>
<th>pval</th>
<th>padj</th>
<th>effect</th>
<th>nSet</th>
<th>nFound</th>
<th>signed</th>
</tr>
</thead>
<tbody>
<tr>
<td>set1</td>
<td>anticorrelated</td>
<td>0.03960396</td>
<td>0.07920792</td>
<td>-18.5941433</td>
<td>40</td>
<td>40</td>
<td>TRUE</td>
</tr>
<tr>
<td>set4</td>
<td>correlated</td>
<td>0.03960396</td>
<td>0.07920792</td>
<td>1.7435310</td>
<td>39</td>
<td>39</td>
<td>TRUE</td>
</tr>
<tr>
<td>set3</td>
<td>correlated</td>
<td>0.12871287</td>
<td>0.17161716</td>
<td>1.9710559</td>
<td>40</td>
<td>40</td>
<td>TRUE</td>
</tr>
<tr>
<td>set2</td>
<td>anticorrelated</td>
<td>0.44554455</td>
<td>0.44554455</td>
<td>-0.1876935</td>
<td>39</td>
<td>39</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>gene.5</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>gene.37</td>
<td>0.55569105</td>
<td>0.484890061</td>
<td>1.1029639</td>
<td>-4.2226726</td>
<td>-1.690514</td>
<td>-3.0980189</td>
</tr>
<tr>
<td>gene.60</td>
<td>-0.30577334</td>
<td>-0.816284729</td>
<td>1.2852544</td>
<td>-2.8099300</td>
<td>-3.614972</td>
<td>-0.9834118</td>
</tr>
<tr>
<td>gene.66</td>
<td>-0.56709908</td>
<td>1.486837939</td>
<td>0.2300383</td>
<td>-2.2423362</td>
<td>-1.837553</td>
<td>-0.8539422</td>
</tr>
<tr>
<td>gene.106</td>
<td>0.06942784</td>
<td>-0.921643964</td>
<td>-1.2380811</td>
<td>-0.3791616</td>
<td>-2.821877</td>
<td>-1.4603829</td>
</tr>
<tr>
<td>gene.108</td>
<td>-0.03259198</td>
<td>1.315295427</td>
<td>-0.4736140</td>
<td>-2.1384493</td>
<td>-1.940030</td>
<td>-1.3709854</td>
</tr>
</tbody>
</table>

Expression scores for each gene set are now available in the geneScores cmapResults column, 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).

```r
def heatmap(set1.expr, scale="none", Colv=NA, labCol=predictor,
            RowSideColors=ifelse( attr(set1.expr, "sign") == "up", "red", "blue"),
            margin=c(7,5))
def 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 a subset of the original query ExpressionSet. As genes can be part of many different gene 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.

```r
> res <- featureScores(cmap, y)
> class(res)
[1] "list"
> names(res)
[1] "set1" "set2" "set3" "set4"
> identical( res[["set1"]], set1.expr )
[1] TRUE
```

R version 3.3.1 (2016-06-21)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.1 LTS

date: 2014-12-02
License: GPL (>= 2)
```
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=en_US.UTF-8 LC_NAME=C
```

13
attached base packages:
[1] stats4  parallel  stats  graphics  grDevices  utils  datasets  base

other attached packages:
[1] DESeq_1.26.0  lattice_0.20-34  locfit_1.5-9.1
[4] gMAP_1.18.0  limma_3.30.0  GSEABase_1.36.0
[7] graph_1.52.0  annotate_1.52.0  XML_3.98-1.4
[10] AnnotationDbi_1.36.0  IRanges_2.8.0  S4Vectors_0.12.0
[13] Biobase_2.34.0  BiocGenerics_0.20.0

loaded via a namespace (and not attached):
[1] splines_3.3.1  GSEAlm_1.34.0  xtable_1.8-2  Category_2.40.0
[5] tools_3.3.1  grid_3.3.1  DBI_0.5-1  genefilter_1.56.0
[9] survival_2.39-5  RBGL_1.50.0  Matrix_1.2-7.1  RColorBrewer_1.1-2
[13] geneplotter_1.52.0  bitops_1.0-6  RCurl_1.95-4.8  RSQLite_1.0.0