Gene set analyses with the gCMAP package

Thomas Sandmann and Richard Bourgon

October 29, 2019

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

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 `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.

```
> 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 `GeneSetCollec-tions` , 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, 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'.

    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
... (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        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

> 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

> 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 Expression-Set 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 containing 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 0.04950495 0.0990099 -18.1394626   40     40   TRUE
2 set4      correlated 0.04950495 0.0990099   2.8394696   39     39   TRUE
3 set3 anticorrelated 0.17821782 0.2376238  -1.0283187   40     40   TRUE
4 set2      correlated 0.41584158 0.4158416   0.5457554   40     40   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 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..

   set trend   pval      padj nSet signed
1 set3  Down 0.0660 0.2630000   40  FALSE
2 set4    Up 0.4060 0.7136667   39  FALSE
3 set1    Up 0.5355 0.7136667   40  FALSE
4 set2  Down 0.7785 0.7785000   40  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, 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.

   set trend      pval      padj       effect nSet nFound signed
1 set3  Down 0.1518880 0.6075520 -0.017673733   40     40  FALSE
2 set4    Up 0.5120371 0.7974286 -0.006552903   39     39  FALSE
3 set1    Up 0.6370618 0.7974286  0.003749492   40     40  FALSE
```

```
4 set2  Down 0.7974286 0.7974286 -0.007800256   40     40  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 0.0003 0.00360   40     40  FALSE
2 set3  Down 0.0251 0.15060   40     40  FALSE
3 set4    Up 0.2369 0.73890   39     39  FALSE
4 set2  Down 0.2463 0.73890   40     40  FALSE
5 set1    Up 0.3573 0.85752   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      correlated 0.01322858 0.05291433  2.47755549   40     40   TRUE
2 set1 anticorrelated 0.50258109 0.70089216 -0.67043412   40     40   TRUE
3 set4 anticorrelated 0.52566912 0.70089216 -0.63463098   39     39   TRUE
4 set2 anticorrelated 0.93556014 0.93556014 -0.08085139   40     40   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      correlated 0.06368996 0.2547598  1.52451580   40     40   TRUE
2 set4 anticorrelated 0.22921011 0.4584202 -0.74145070   39     39   TRUE
3 set2      correlated 0.40517372 0.4718261  0.23997784   40     40   TRUE
4 set1 anticorrelated 0.47182609 0.4718261 -0.07068032   40     40   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 set3    up  1.0000000   40     40   TRUE
2 set4  down -1.0000000   39     39   TRUE
3 set2    up  0.9169139   40     40   TRUE
4 set1    up  0.0000000   40     40   TRUE
... (only top 5 results shown, use 'cmapTable' function to see all) ...
```
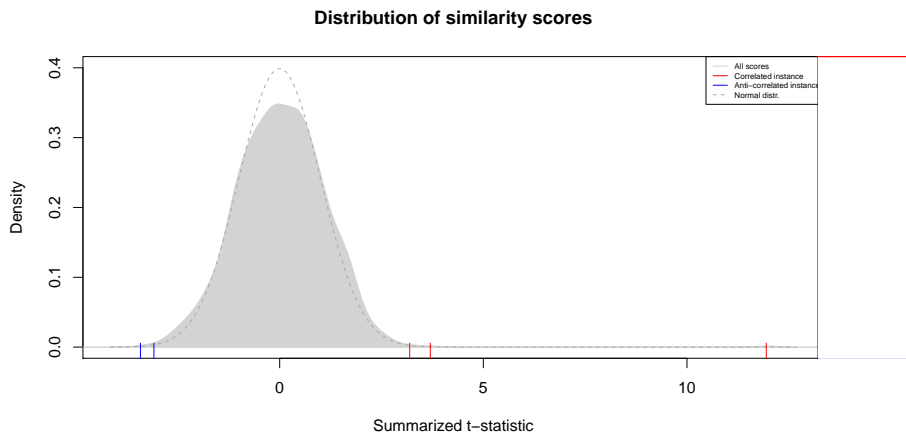
6 Overview plots

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.
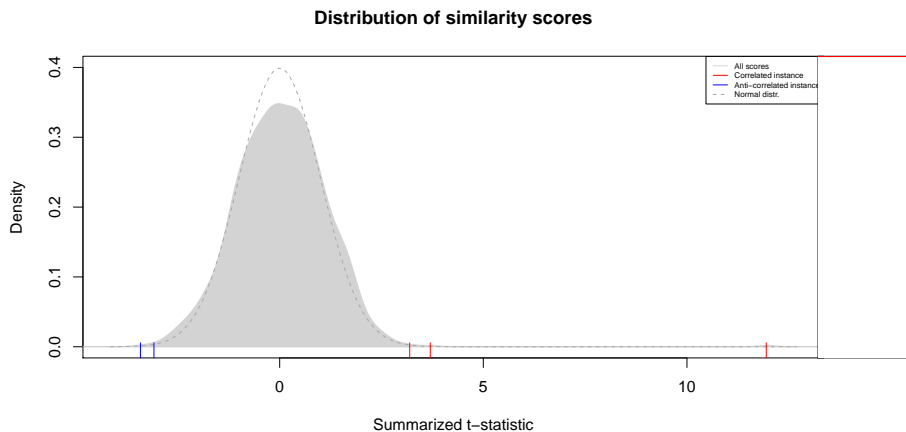
```
> ## create random score profile
> set.seed(123)
> z <- rnorm(1000)
> names(z) <- paste("g", 1:1000, sep="")
> ## generate random incidence matrix of gene sets
> n <-replicate(1000, {
+    s <- rep(0,1000)
+    s[ sample(1:1000, 20)] <-  1
+    s[ sample(1:1000, 20)] <- -1
+    s
+    })
> dimnames(n) <- 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 6.698361e-33 6.698361e-30 11.947406   40     40   TRUE
2 set941      correlated 2.174263e-04 1.087131e-01  3.697858   40     40   TRUE
3 set648  anticorrelated 6.287339e-04 2.095780e-01 -3.418906   40     40   TRUE
4 set664      correlated 1.411126e-03 3.527814e-01  3.192366   40     40   TRUE
5 set732  anticorrelated 2.015862e-03 4.031725e-01 -3.087885   40     40   TRUE
... (only top 5 results shown, use 'cmapTable' function to see all) ...

> plot(res)
```

Distribution of similarity scores

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.)

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

Distribution of similarity scores

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 0.06930693 0.1386139 -18.1394626   40     40   TRUE
2 set4     correlated 0.06930693 0.1386139   2.8394696   39     39   TRUE
3 set3 anticorrelated 0.27722772 0.3696370  -1.0283187   40     40   TRUE
4 set2     correlated 0.45544554 0.4554455   0.5457554   40     40   TRUE
     geneScores
1 40 x 6 matrix
2 39 x 6 matrix
3 40 x 6 matrix
4 40 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.1   -0.4676850 -0.9164483  0.04133489 -1.649489 -2.1038079 -3.5465618
gene.52  -0.7017887  0.7063692  0.20367207  1.048797  0.6993498  0.1803723
gene.103 -0.5893912  0.9316614  0.95587988 -1.288340 -1.4181353 -2.8523997
gene.113 -0.7683347 -0.8633709  0.58939571 -1.668827 -2.2778195 -2.0796107
gene.124 -0.8306497 -2.7617287 -0.56187130 -2.488218 -0.3598665 -3.2238847
gene.134  1.7222775 -1.3034488 -0.62093701  1.112516  1.9062430  1.6451637
```
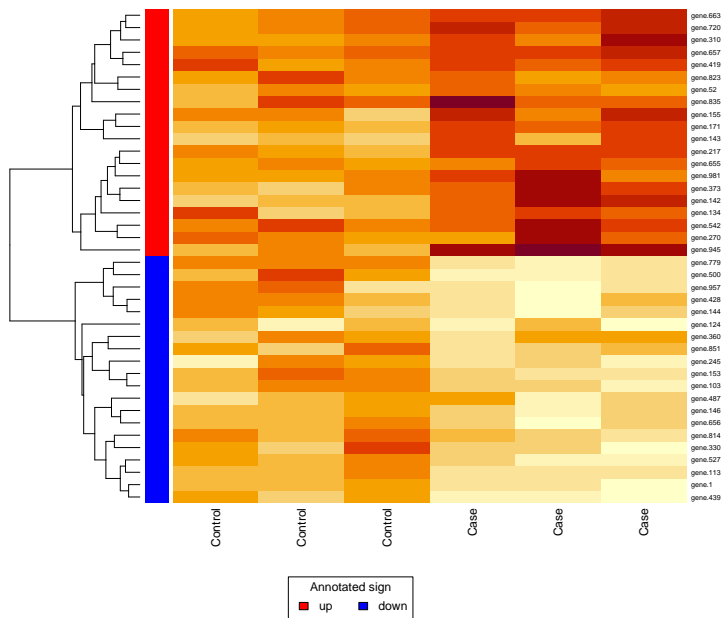
Expression scores for each gene set are now available in the geneScores `cmapResults` colum, which can be accessed through a method with the same name. Each matrix of expression scores is accompanied by an additional 'sign' attribute to remind us whether gene set members were annotated as up- or down-regulated.

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

>   sessionInfo()

R version 3.6.1 (2019-07-05)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.04.3 LTS

Matrix products: default
BLAS:   /home/biocbuild/bbs-3.10-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.10-bioc/R/lib/libRlapack.so

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
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C


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


other attached packages:
 [1] bigmemory_4.5.33    DESeq_1.38.0        lattice_0.20-38
 [4] locfit_1.5-9.1      gCMAP_1.30.0        limma_3.42.0
 [7] GSEABase_1.48.0     graph_1.64.0        annotate_1.64.0
[10] XML_3.98-1.20       AnnotationDbi_1.48.0 IRanges_2.20.0
[13] S4Vectors_0.24.0    Biobase_2.46.0      BiocGenerics_0.32.0


loaded via a namespace (and not attached):
 [1] Rcpp_1.0.2          RColorBrewer_1.1-2   pillar_1.4.2
 [4] compiler_3.6.1      bigmemoryExtras_1.34.0 bitops_1.0-6
 [7] tools_3.6.1         zeallot_0.1.0        digest_0.6.22
[10] bit_1.1-14          RSQLite_2.1.2        memoise_1.1.0
[13] tibble_2.1.3        pkgconfig_2.0.3      rlang_0.4.1
[16] Matrix_1.2-17       DBI_1.0.0            Category_2.52.0
[19] GSEAlm_1.46.0       genefilter_1.68.0    vctrs_0.2.0
[22] bit64_0.9-7         grid_3.6.1          bigmemory.sri_0.1.3
[25] RBGL_1.62.0         survival_2.44-1.1    geneplotter_1.64.0
[28] blob_1.2.0          splines_3.6.1       backports_1.1.5
[31] xtable_1.8-4        RCurl_1.95-4.12      crayon_1.3.4
```