

cnvGSA: Gene-Set Analysis of (Rare) Copy Number Variants

Daniele Merico and Robert Ziman
The Centre for Applied Genomics
daniele.merico@gmail.com, robert.ziman@gmail.com

March 31, 2012

Contents

1	Overview	2
2	Workflow outline	2
3	Loading input data	4
3.1	Loading CNV data	4
3.1.1	Loading <code>cnvData</code> from files	5
3.2	Loading gene sets	7
3.2.1	Loading gene-sets from a <code>.gmt</code> file	8
3.2.2	Sources of gene-sets	8
3.3	Loading gene annotations	8
3.3.1	Loading annotations from the ‘ <code>org.Hs.eg.db</code> ’ Bioconductor package	9
4	Configuring test parameters	9
4.1	Loading test parameters from a file	10
5	Running the association test	11
6	Full workflow example: case-control analysis of rare CNVs from the Pinto et al. 2010 ASD study	11
6.1	Loading the data and running the association test	11
6.2	Reviewing the results	14
6.2.1	Enrichment results and gene-centric statistics	14
6.2.2	Detailed analysis of gene-set associations	19
6.2.3	Burden analysis	20
7	References	23

1 Overview

`cnvGSA` is an R package meant to facilitate gene-set analysis of (rare) copy number variants (CNVs).

Known gene-sets are tested for prevalence of rare variants in case vs. control subjects. Whenever a subject has at least one gene in a gene-set affected by a rare variant, a perturbation count of 1 is assigned to the (subject, gene-set) pair; for each gene-set, subject counts are tested vs. control counts using the Fisher Exact Test (FET). Significant gene-sets will have a significantly high count in cases compared to controls. Statistical reports on burden are also generated.

Note: this analysis requires that subjects be unrelated and that case/control cohorts be matched by sex, age, ethnicity, and other potential confounders (such as platform and CNV detection methods). In addition, only rare CNVs should be present. The definition of ‘rare’ is typically based on frequency in the study subjects or on a larger data-set of independent controls that are used to remove putative common regions – or on a combination of both techniques. For more details about this, see [2] and [3].

2 Workflow outline

The general procedure for performing a CNV gene-set analysis involves loading CNVs and gene-sets and related data, setting filters and parameters, running the analysis, and reviewing the results. To facilitate these operations, the package provides `"CnvGSAInput"`, an S4 class acting as a simple container data structure with slots for each required input data structure:

```
> library("cnvGSA")
> slotNames("CnvGSAInput")

[1] "cnvData" "gsData" "geneData" "params"
```

The input slots should hold the following:

- `cnvData` - CNV data
- `gsData` - Gene-set data
- `geneData` - Gene annotations (symbols and descriptive names)
- `params` - Test parameters

To ease the discussion here, a pre-built input object has been saved for convenience in the companion data package for this vignette:

```
> library("cnvGSAdata")
> data("cnvGSA_input_example")
> ls()
```

```

[1] "input"

> class(input)

[1] "CnvGSAInput"
attr(,"package")
[1] "cnvGSA"

> slotNames(input)

[1] "cnvData" "gsData" "geneData" "params"

```

Each of the slots can be accessed using an accessor function of the same name (e.g. `cnvData(input)` gets or sets `cnvData`, `gsData(input)` gets or sets `gsData`, etc.).

The input object is used with `cnvGSA`'s main function, `cnvGSAFisher()`:

- `cnvGSAFisher(input)` - Performs a gene-set association test of case vs. control subjects using the Fisher Exact Test.

This function produces as its output an object of class `"CnvGSAOutput"` – likewise a simple S4 class that has slots for each output data structure.

```

> data("cnvGSA_output_example")
> ls()

[1] "input" "output"

> class(output)

[1] "CnvGSAOutput"
attr(,"package")
[1] "cnvGSA"

> slotNames(output)

[1] "cnvData" "burdenSample" "burdenGs" "geneData" "enrRes"

```

The output slots contain the following:

- `cnvData` - Original and filtered CNV data
- `burdenSample` - Burden analysis results for subjects
- `burdenGs` - Burden analysis results for gene-sets
- `geneData` - Gene-centric statistics
- `enrRes` - Gene-set enrichment results

As with the slots in the input object, each of these can likewise be accessed using an accessor function of the same name (`burdenSample()` gets `burdenSample`, etc.).

Throughout the following sections, the pre-built `input` example object from above will be shown first to illustrate its structure; each of its elements will then be recreated with the suffix “_demo” to demonstrate the syntax of the functions used to load them. Section 5 then shows how to rebuild the full input object using the `CnvGSAInput` constructor and then run the association test, and section 6 shows the full workflow example (i.e. all the code in a single listing) along with a detailed discussion of how to review and interpret the results.

3 Loading input data

3.1 Loading CNV data

The first input data structure that needs to be loaded is `cnvData`. As mentioned earlier, the `cnvData()` function below is just an accessor for this slot:

```
> str( cnvData(input), strict.width="cut" )
```

List of 4

```
$ cnv      : 'data.frame':      5478 obs. of  7 variables:
 ..$ SampleID: chr [1:5478] "1020_4" "1020_4" "1020_4" "1030_3" ...
 ..$ Chr      : chr [1:5478] "3" "4" "6" "7" ...
 ..$ Coord_i  : int [1:5478] 4110452 34802932 35606076 64316996 56265896 39957..
 ..$ Coord_f  : int [1:5478] 4145874 35676439 35673400 64593616 56361311 40082..
 ..$ Type     : chr [1:5478] "DEL" "DUP" "DUP" "DEL" ...
 ..$ Genes    : chr [1:5478] "" "" "2289" "168374" ...
 ..$ CnvID    : chr [1:5478] "CNV_1" "CNV_2" "CNV_3" "CNV_4" ...
$ s2class: 'data.frame':      2035 obs. of  2 variables:
 ..$ Class    : chr [1:2035] "case" "case" "case" "case" ...
 ..$ SampleID: chr [1:2035] "1020_4" "1030_3" "1045_3" "1050_3" ...
$ gsep      : chr ";"
$ filters: List of 1
 ..$ limits_type: chr "DEL"
```

Its elements are as follows:

- `$cnv` - A data frame containing the CNVs. Each row contains data for one CNV:
 - `SampleID` - ID assigned to the subject’s DNA sample in which the CNV was found. The values here should match the corresponding values in the `$s2class` data frame (see below).
 - `Chr` - Chromosome on which the CNV is located.

- `Coord_i` - Start position of the CNV on the chromosome.
 - `Coord_f` - End position of the CNV on the chromosome.
 - `Type` - CNV type (typically "DEL" or "DUP").
 - `Genes` - Genes affected by the CNV, stored in a delimited format inside a character string; e.g. "54777;255352;84435" for semicolon-delimited EntrezGene identifiers. (We recommend using this ID system – and in any case the example data in this vignette follows it.) CNVs that are not genic should have an empty string (i.e. "") in this column.
 - `CnvID` - ID assigned to the CNV. (Note that this is also of type `character`.)
- `$s2class` - A data frame with columns `$SampleID` and `$Class` that is used as a lookup table for the sample-to-class mapping. In the current implementation, only two classes are allowed (typically each sample will be of class "case" or "ctrl").
 - `$gsep` - The character used as delimiter in `cnvGenes`.
 - `$filters` - List whose elements are parameters to filter the variants. Possible elements are:
 - `limits_type` (*optional*) - Type of variant to be kept (e.g. "DEL" or "DUP").
 - `limits_size` (*optional; overrides \$limits_type*) - A data frame with the columns:
 - `Type` - Type of variant to be kept
 - `Max_length` - Maximum length of each CNV
 - `Max_gcount` - Maximum number of genes hit by each CNV
 - `rem_genes` (*optional*) - Vector of gene IDs to be removed from the analysis (the variants hitting such genes will be removed as well).

3.1.1 Loading `cnvData` from files

The package provides functions for importing data from files into the various elements of the `cnvData` structure.

- `readGVF(filename)` - Imports CNV data from a `.gvf` (Genome Variation Format) file such as those that can be downloaded from the Database of Genomic Variants (<http://projects.tcag.ca/variation/>). For more information on the `.gvf` file format, see the GVF specification:

Genome Variation Format 1.06

<http://www.sequenceontology.org/resources/gvf.html>

`readGVF()` can be used to load the `cnv` element of the `cnvData` slot.

```
> cnvData_demo <- list()
> cnvFile <- system.file( "extdata", "cnv.gvf", package="cnvGSAdata" )
> cnvData_demo$cnv <- readGVF( cnvFile )
> rm(cnvFile)
> head(cnvData_demo$cnv)
```

SampleID	Chr	Coord_i	Coord_f	Type	Genes	CnvID
1	1020_4	3	4110452	4145874	DEL	CNV_1
2	1020_4	4	34802932	35676439	DUP	CNV_2
3	1020_4	6	35606076	35673400	DUP	CNV_3
4	1030_3	7	64316996	64593616	DEL	CNV_4
5	1030_3	10	56265896	56361311	DEL	CNV_5
6	1045_3	1	39957035	40082808	DUP	CNV_6

Notice that the `Genes` column is empty. To load this column, use `getCnvGenes()`:

- `getCnvGenes(cnv, genemap, delim)` - Takes as input a data frame of CNVs (`cnvData$cnv` can be used directly here), a data frame of gene coordinates and returns the genes hit by each CNV. The output is a vector – which can be directly assigned to `cnvGenes` – in which each element contains a delimited string of the genes falling within the range of the corresponding CNV in the input.

The `cnvGSAdata` package contains a pair of GFF files that can be used to load the `genemap` data frame. Shown below is the code for building this data frame from the file containing exon coordinates (note that the runtime of `getCnvGenes()` may be on the order of 10-20 minutes for a full lookup of CNV genes using these examples):

```
> genemapFile <- system.file(
+   "extdata",
+   "merge_00k_flank_hg18_refGene_jun_2011_exon.gff",
+   package = "cnvGSAdata"
+ )
> fields <- read.table (
+   genemapFile,
+   sep = "\t",
+   comment.char = "",
+   quote = "\"",
+   header = FALSE,
+   stringsAsFactors = FALSE
+ )
> genemap <- data.frame(
+   Chr = fields[,1],
+   Coord_i = fields[,4],
+   Coord_f = fields[,5],
+   GeneID = fields[,11],
+   stringsAsFactors = FALSE
+ )
> genemap$Chr <- sub( genemap$Chr, pattern = "chr", replacement = "" )
> cnvData_demo$gsep <- ";"
> cnvData_demo$cnv$Genes <- getCnvGenes( cnv=cnvData$cnv, genemap=genemap,
```

```
+   delim=cnvData_demo$gsep )
> rm( genemapFile, fields, genemap )
```

The `$s2class` element can be loaded from a file in a straightforward way using R's standard `read.table()` function:

```
> s2classFile <- system.file( "extdata", "s2class.txt", package="cnvGSAdata" )
> cnvData_demo$s2class <- read.table(
+   s2classFile,
+   sep = "\t",
+   col.names = c("SampleID", "Class"),
+   stringsAsFactors = FALSE
+ )
> rm(s2classFile)
```

Finally, `$filters` is loaded along with the main test parameters (see section 4).

3.2 Loading gene sets

The next data structure that needs to be loaded is `gsData`, which contains the gene-set data. Again, the `gsData()` function below is just an accessor function for this slot:

```
> str( gsData(input), list.len=4 )
```

List of 2

```
$ gs2gene:List of 3722
..$ G0:0030850: chr [1:42] "2736" "5176" "9241" "8626" ...
..$ G0:0030856: chr [1:34] "595" "54206" "8626" "4435" ...
..$ G0:0030855: chr [1:206] "56033" "2302" "3713" "353142" ...
..$ G0:0031100: chr [1:40] "890" "4899" "578" "595" ...
.. [list output truncated]
$ gs2name: Named chr [1:3722] "prostate gland development" "regulation of epithelial ce
..- attr(*, "names")= chr [1:3722] "G0:0030850" "G0:0030856" "G0:0030855" "G0:0031100"
```

The list elements are:

- `$gs2gene` - A list of character vectors where each vector contains the genes for a particular gene-set; the gene-set names ("G0:0030850" etc.) are stored as the **names** of the list elements. Since gene-sets can hold different numbers of genes, the vectors will typically have different lengths.
- `$gs2name` - A single character vector mapping each gene-set name to its description. The descriptions are stored as the vector elements and the gene-set names are stored as the **names** of the vector elements.

3.2.1 Loading gene-sets from a .gmt file

The package provides a function for loading gene-sets directly from files:

- `readGMT(filename)` - Imports gene-set data from a .gmt (Gene Matrix Transposed) file such as those that can be downloaded from the GSEA/MSigDB database. For more information on MSigDB and the .gmt file format, see:

MSigDB: Molecular Signatures Database
<http://www.broadinstitute.org/gsea/msigdb/index.jsp>

GSEA wiki: Data formats
http://www.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats

```
> gsDataFile <- system.file( "extdata", "gsData.gmt", package="cnvGSAdata" )
> gsData_demo <- readGMT( gsDataFile )
> rm(gsDataFile)
```

3.2.2 Sources of gene-sets

The first few entries in the `gsData` example above (page 7) show gene-set data collected from the Gene Ontology database (<http://www.geneontology.org/>). Gene-sets can also be derived from other public databases such as:

- PFAM: <http://pfam.sanger.ac.uk/>
- NCI: <http://cactus.nci.nih.gov/ncidb2.1/>
- KEGG: <http://www.genome.jp/kegg/>
- Biocarta: <http://www.biocarta.com/genes/index.asp>
- Reactome: <http://www.reactome.org/>

3.3 Loading gene annotations

The `geneData` slot in the input should contain the gene annotations:

```
> str( geneData(input), strict.width="cut" )
```

```
List of 1
```

```
$ ann:List of 2
```

```
..$ gene2sy : Named chr [1:44811] "A1BG" "NAT2" "ADA" "CDH2" ...
.. ..- attr(*, "names")= chr [1:44811] "1" "10" "100" "1000" ...
..$ gene2name: Named chr [1:44811] "alpha-1-B glycoprotein" "N-acetyltransfe..
.. ..- attr(*, "names")= chr [1:44811] "1" "10" "100" "1000" ...
```

Note that the annotations are stored in the intermediate list `$ann`. It contains two character vectors: the first has the gene symbols; the second has the full descriptive names of the genes.

3.3.1 Loading annotations from the 'org.Hs.eg.db' Bioconductor package

These two vectors can be loaded quickly and easily using the `org.Hs.eg.db` Bioconductor package, e.g. as in following code:

```
> library( "org.Hs.eg.db" )
> ann <- list( gene2sy = character(0), gene2name = character(0) )
> x <- org.Hs.egSYMBOL
> mapped_genes <- mappedkeys(x)
> ann$gene2sy <- unlist( as.list( x[mapped_genes] ) )
> x <- org.Hs.egGENENAME
> mapped_genes <- mappedkeys(x)
> ann$gene2name <- unlist( as.list( x[mapped_genes] ) )
> geneData_demo <- list(ann)
> rm( ann, x, mapped_genes )
```

4 Configuring test parameters

The main association test procedure accepts several parameters as a simple list (note that the `params()` function below is just an accessor function for this slot):

```
> str( params(input) )
```

List of 6

```
$ grandtotals_mode: chr "all"
$ sample_classes  : chr [1:2] "case" "ctrl"
$ fdr_iter        : num 2
$ extended_report : num 200
$ filters         :List of 1
  ..$ limits_type: chr "DEL"
$ boxplot_PDFs   : logi FALSE
```

The parameters are as follows:

- `grandtotals_mode` - Used to modify the grand totals in the FET; should be set depending on the results of the burden analysis. Possible values are:
 - "all" - Produce totals using all samples in the study
 - "cnv" - Produce totals using filtered samples hit by at least one variant
 - "cnvGen" - Produce totals using filtered samples hit by at least one genic variant
- `sample_classes` - The sample classes, e.g. "case" and "ctrl".

- `fdr_iter` - The number of iterations to perform in the empirical FDR (False Discovery Rate) estimation done by randomizing sample class (i.e. case, control) assignments. (Note: larger gene-sets have more statistical power; this can be taken into account by separately testing gene-sets with different sizes.)
- `extended_report` - The number of gene-sets for which the extended report will be generated. (The extended report is the ‘`extended`’ structure in the output; see section 6.2.1.)
- `filters` - A list of parameters for filtering the CNVs (see section 3.1).
- `boxplot_PDFs` - Boolean indicating whether or not to produce PDFs containing boxplots of the burden statistics (cf. section 6.2.3).

4.1 Loading test parameters from a file

To make it easier to integrate the association test into a larger bioinformatics pipeline, it is convenient to read in the parameters from an external source such as a text file. One such implementation is to record each parameter on its own line using R syntax:

```
# Main test parameters
grandtotals_mode <- "all"
sample_classes  <- c( "case", "ctrl" )
fdr_iter        <- 1000
extended_report <- 200
boxplot_PDFs   <- FALSE

# cnvData$filters parameters
limits_type     <- "DEL"
```

The package provides a simple function to parse such a file (essentially just `source()`ing it and then handling the few possibilities around the `$filters` parameters):

- `readParamsRFile(filename)` - Read test parameters from `filename` and return a list object that can be passed to the `params` argument in `cnvGSAFisher()`. The file is assumed to have all the main test parameters as above. For the `$filters` parameters (cf. section 3), `$limits_type` and `$rem_genes` can be assigned directly (as `$limits_type` is in the example above); `$limits_size` should be specified by assigning its elements `$Type`, `$Max_length`, and `$Max_gcount`.

```
> paramFile <- system.file( "scripts", "params_example.R", package="cnvGSA" )
> params_demo <- readParamsRFile( paramFile )
> rm(paramFile)
```

At this point the `$filters` element of `cnvData` can be assigned:

```
> cnvData_demo$filters <- params_demo$filters
```

5 Running the association test

Now that each of its individual elements have been loaded, the input object can be built using the `CnvGSAInput()` constructor:

```
> input_demo <- CnvGSAInput(  
+   cnvData = cnvData_demo,  
+   gsData = gsData_demo,  
+   geneData = geneData_demo,  
+   params = params_demo  
+ )  
> rm( cnvData_demo, gsData_demo, geneData_demo, params_demo )
```

The association test can now be run by calling the main function.

```
> output <- cnvGSAFisher( input )
```

Note that a high `fdr_iter` will require a noticeable runtime. For example, on a typical desktop workstation setup as of early 2012, the runtime for an input dataset similar to the full workflow example (5500 CNVs (averaging 1 gene per filtered CNV) against 3700 gene-sets; see section 6), with an `fdr_iter` of 1000, will be on the order of 1 hour.

(Also note that if you are running the test more than once with the same input and parameters, be sure to first call `set.seed()` so that the random number generator is consistent each time; otherwise the FDR calculation will be slightly different for each run.)

6 Full workflow example: case-control analysis of rare CNVs from the Pinto et al. 2010 ASD study

6.1 Loading the data and running the association test

The following code performs an analysis of 5500 CNVs against 3700 gene-sets using an `fdr_iter` of 1000. The CNVs are rare CNVs from ASD consortium data (Pinto et al., Nature 2010 [1]) and the gene-sets are a combination of those from the Gene Ontology, KEGG, Biocarta, Reactome, and PFAM (see section 3.2).

```
library( "cnvGSA" )  
library( "cnvGSAdata" )  
library( "org.Hs.eg.db" )    ## for gene annotations
```

```
##  
## Load data and parameters
```

```

##

## CNVs
cnvData <- list()
cnvFile <- system.file( "extdata", "cnv.gvf", package="cnvGSAdata" )
cnvData$cnv <- readGVF( cnvFile )
rm(cnvFile)

## CNV genes
## (N.B. may take several minutes to run the full example CNV data against
## the full example gene map)
genemapFile <- system.file(
  "extdata",
  "merge_00k_flank_hg18_refGene_jun_2011_exon.gff",
  package = "cnvGSAdata"
)
fields <- read.table (
  genemapFile,
  sep = "\t",
  comment.char = "",
  quote = "\"",
  header = FALSE,
  stringsAsFactors = FALSE
)
genemap <- data.frame(
  Chr = fields[,1],
  Coord_i = fields[,4],
  Coord_f = fields[,5],
  GeneID = fields[,11],
  stringsAsFactors = FALSE
)
genemap$Chr <- sub( genemap$Chr, pattern = "chr", replacement = "" )
cnvData$gsep <- ";"
cnvData$cnv$Genes <- getCnvGenes(
  cnv = cnvData$cnv,
  genemap = genemap,
  delim = cnvData$gsep
)
rm( genemapFile, fields, genemap )

## Sample classes
s2classFile <- system.file( "extdata", "s2class.txt", package="cnvGSAdata" )
cnvData$s2class <- read.table(

```

```

    s2classFile,
    sep = "\t",
    col.names = c("SampleID", "Class"),
    stringsAsFactors = FALSE
)
rm(s2classFile)

## Gene sets
gsDataFile <- system.file( "extdata", "gsData.gmt", package="cnvGSAdata" )
gsData <- readGMT( gsDataFile )
rm(gsDataFile)

## Gene annotations
ann <- list( gene2sy = character(0), gene2name = character(0) )
x <- org.Hs.egSYMBOL
mapped_genes <- mappedkeys(x)
ann$gene2sy <- unlist( as.list( x[mapped_genes] ) )
x <- org.Hs.egGENENAME
mapped_genes <- mappedkeys(x)
ann$gene2name <- unlist( as.list( x[mapped_genes] ) )
geneData <- list(ann)
rm( ann, x, mapped_genes )

## Parameters
paramFile <- system.file( "scripts", "params_example.R", package="cnvGSA" )
params <- readParamsRFile( paramFile )
cnvData$filters <- params$filters
rm( paramFile )

##
## Create the input object
##

input <- CnvGSAInput(
  cnvData = cnvData,
  gsData = gsData,
  geneData = geneData,
  params = params
)
rm( cnvData, gsData, geneData, params )

```

```
##
## Run association test and save the output
##

output <- cnvGSAFisher( input )

save( output, file = "cnvGSA_output_example.RData" )
```

6.2 Reviewing the results

(Note: Since the runtime for the full workflow example above, with `fdr_iter` of 1000, may take on the order of one hour or more on a modern workstation (cf. section 5), we have included the saved output in the companion data package as shown in the workflow outline section.)

As stated in the workflow outline, the output object is a simple S4 class containing a slot for each output data structure:

```
> slotNames(output)

[1] "cnvData"      "burdenSample" "burdenGs"     "geneData"     "enrRes"
```

Similar to the input, each of these is a list structure containing further data structures: `cnvData` contains the original and filtered CNV data, `enrRes` contains the gene-set enrichment results, and `burdenSample`, `burdenGs`, and `geneData` contain burden analysis and gene-centric statistics that can be used to ensure the validity of the enrichment results. Just as with the input object, each of these can be accessed using an accessor function of the same name.

6.2.1 Enrichment results and gene-centric statistics

Taking a look first at `enrRes`:

```
> str( enrRes(output), max.level=1 )

List of 4
 $ basic   :'data.frame':      3368 obs. of  16 variables:
 $ totals  : Named int [1:2] 889 1146
 ..- attr(*, "names")= chr [1:2] "case" "ctrl"
 $ extended:'data.frame':      200 obs. of  31 variables:
 $ gstables:List of 200
 .. [list output truncated]
```

The data frames `basic` and `extended` contain the actual enrichment results. `basic` has the results for all affected gene-sets, whereas `extended` has several additional columns of

information but only for the most highly enriched gene-sets (the number of which can be set by the `extended_report` parameter; see section 4). `totals` simply shows the total number of case and control samples. `gstables` contains a list of tables with CNV and gene information specific to each gene-set (this is discussed in more detail in the following section).

The structure of `extended` is shown in the listing below (`basic` has the same structure but only goes up to `FET_permFDR`):

```
> str( enrRes(output)$extended, max.level=1, strict.width="cut" )
```

```
'data.frame':      200 obs. of  31 variables:
 $ GsID           : chr  "G0:0005929" "G0:0030030" "G0:0006928" "G0:000581..
 $ GsName         : chr  "cilium" "cell projection organization" "cellular..
 $ GsSize         : int  170 685 685 39 590 590 343 70 485 554 ...
 $ case_N         : num  37 62 32 16 30 30 25 28 22 28 ...
 $ ctrl_N         : num  11 33 10 1 9 9 6 8 5 9 ...
 $ case_%         : num  4.16 6.97 3.6 1.8 3.37 ...
 $ ctrl_%         : num  0.9599 2.8796 0.8726 0.0873 0.7853 ...
 $ FET_pv         : num  1.99e-06 1.19e-05 1.57e-05 1.64e-05 2.12e-05 ...
 $ FET_OR         : num  4.48 2.53 4.24 20.96 4.41 ...
 $ FET_ORconfLow  : num  2.45 1.73 2.24 3.92 2.26 ...
 $ FET_ORconfHigh : num  Inf Inf Inf Inf Inf ...
 $ FET2s_OR       : num  4.48 2.53 4.24 20.96 4.41 ...
 $ FET2s_ORconfLow : num  2.22 1.61 2.02 3.24 2.03 ...
 $ FET2s_ORconfHigh : num  9.79 4.02 9.72 877.2 10.62 ...
 $ FET_bhFDR      : num  0.00669 0.01188 0.01188 0.01188 0.01188 ...
 $ FET_permFDR    : num  0 0.0015 0.001 0.001 0.000833 ...
 $ Support_size_case : int  13 30 27 1 25 25 18 6 17 23 ...
 $ Support_ratio_case : num  2.23 1.277 1.149 0.748 1.236 ...
 $ Support_geneid_case: chr  "4867;9576;64518;2782;83659;65217;221322;164714;5..
 $ Support_symbol_case: chr  "NPHP1;SPAG6;TEKT3;GNB1;TEKT1;PCDH15;C6orf170;TTL..
 $ Support_size_ctrl : int  4 12 3 1 2 2 3 3 3 2 ...
 $ Support_ratio_ctrl : num  0.843 0.628 0.157 0.919 0.121 ...
 $ Support_geneid_ctrl: chr  "51626;51057;27241;84075" "152273;1287;775;57689;..
 $ Support_symbol_ctrl: chr  "DYNC2LI1;C2orf86;BBS9;FSCB" "FGD5;COL4A5;CACNA1C..
 $ case_SampleID   : Factor w/ 157 levels "1030_3;1128_3;1199_3;1265_8;1303..
 $ ctrl_SampleID   : Factor w/ 112 levels "B106672_1007874643;B187727_00679..
 $ case_CnvID      : Factor w/ 159 levels "CNV_101;CNV_151;CNV_335;CNV_1211..
 $ ctrl_CnvID      : Factor w/ 114 levels "CNV_2399;CNV_2620;CNV_2639;CNV_2..
 $ FETpv_remTop    : num  9.74e-03 5.68e-03 4.97e-05 1.00 6.82e-05 ...
 $ FETfdr_remTop   : List of 200
 .. [list output truncated]
 $ Topgene         : chr  "CROCC" "CROCC" "ERBB4" "CROCC" ...
```

Each row contains results for a single gene-set. The columns are as follows:

- **GsID**, **GsName**, and **GsSize** show the gene-set’s identifier, name, and number of member genes respectively.
- **case_N/ctrl_N** and **case_%/ctrl_%** show the number and percentage of case and control samples hitting the gene-set.
- **FET_pv**, **FET_OR**, **FET_ORconfLow**, and **FET_ORconfHigh** show the p-value, odds ratio, and low and high bounds of the confidence interval for the **one-sided** Fisher Exact Test of the gene-set.
- **FET2s_pv**, **FET2s_OR**, **FET2s_ORconfLow**, and **FET2s_ORconfHigh** show the p-value, odds ratio, and low and high bounds of the confidence interval for the **two-sided** Fisher Exact Test of the gene-set.
- **FET_bhFDR** and **FET_permFDR** show the Benjamini-Hochberg corrected p-value and the permutation-based FDR, respectively.
- **Support_size_case**, **Support_ratio_case**, **Support_geneid_case**, and **Support_symbol_case** show the number, ratio, IDs, and symbols of the “case support genes” involving only the genes from this gene-set. “Case support genes” are defined as those genes whose counts (over all samples in the study) are greater in cases than in controls. **Support_ratio_case** accordingly shows the ratio of (the number of case support genes for this gene-set in particular) to (the total number of case support genes over all gene-sets in the study). (The set of case support genes over all gene-sets is provided in **geneData** in the output; see the description a little further down in this section.)
- **Support_size_ctrl**, **Support_ratio_ctrl**, **Support_geneid_ctrl**, and **Support_symbol_ctrl** show the corresponding “control support genes” (i.e. same as above but with genes whose counts are greater in controls than in cases).
- **FETpv_remTop** and **FETfdr_remTop** show the exact and permuted p-values when the top associated gene in the gene-set is removed.
- **Topgene** shows the top associated gene in the gene-set.

As mentioned above, the **basic** data frame contains only those columns going up to **FET_permFDR** – but for all gene-sets in the study. This is sufficient to identify those gene-sets passing the FDR threshold:

```
> which( enrRes(output)$basic$FET_permFDR <= 0.01 )
```

```
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
```

Note that although the permutation FDR is not necessarily monotonic (that is, when the gene-sets are ranked in order of **FET_pv** – as is the case in the output), this particular output lists all of the top 20 gene-sets as having passed the threshold. Taking a closer look now at **extended** to see their p-values:

```
> head( enrRes(output)$extended[ , c("FET_pv", "FET_OR", "FET_bhFDR", "FET_permFDR",
+ "Topgene")], 20 )
```

	FET_pv	FET_OR	FET_bhFDR	FET_permFDR	Topgene
GO:0005929	1.985592e-06	4.477803	0.006687474	0.0000000000	CROCC
GO:0030030	1.189627e-05	2.527383	0.011878618	0.0015000000	CROCC
GO:0006928	1.569093e-05	4.238933	0.011878618	0.0010000000	ERBB4
GO:0005814	1.643013e-05	20.963648	0.011878618	0.0010000000	CROCC
GO:0048870	2.116143e-05	4.409092	0.011878618	0.0008333333	ERBB4
GO:0051674	2.116143e-05	4.409092	0.011878618	0.0008333333	ERBB4
GO:0007265	2.542393e-05	5.494198	0.011882990	0.0007142857	ARHGEF5
GO:0044441	2.822563e-05	4.622781	0.011882990	0.0007500000	CROCC
GO:0008284	6.025653e-05	5.787094	0.020874673	0.0015555556	ERBB4
GO:0016477	6.817738e-05	4.105704	0.020874673	0.0013636364	ERBB4
GO:0051056	6.817738e-05	4.105704	0.020874673	0.0013636364	ARHGAP11B
GO:0007264	9.328775e-05	3.357964	0.026182762	0.0032500000	ARHGAP11B
GO:0045121	1.419163e-04	6.229265	0.035190752	0.0045384615	ERBB4
GO:0005096	1.671770e-04	7.420950	0.035190752	0.0051250000	ARHGAP11B
GO:0031023	1.671770e-04	7.420950	0.035190752	0.0051250000	CROCC
GO:0051297	1.671770e-04	7.420950	0.035190752	0.0051250000	CROCC
GO:0030695	3.032652e-04	3.017567	0.054020858	0.0098333333	ARHGAP11B
GO:0060589	3.032652e-04	3.017567	0.054020858	0.0098333333	ARHGAP11B
GO:0046578	3.047495e-04	4.593581	0.054020858	0.0100526316	ARHGEF5
GO:0010035	3.307224e-04	6.976431	0.055693653	0.0102500000	ERBB4

The gene-set association p-values above look promising. Note however that just four genes are shown as the top associated genes for these gene-sets; could it be that several of the gene-sets are acquiring most of their association from these (e.g. if they happen to be single highly associated genes)? `geneData` provides statistics that may be helpful here:

```
> str( geneData(output), strict.width="wrap" )
```

```
List of 6
 $ ann :List of 2
 ..$ gene2sy : Named chr [1:44811] "A1BG" "NAT2" "ADA" "CDH2" ...
 .. ..- attr(*, "names")= chr [1:44811] "1" "10" "100" "1000" ...
 ..$ gene2name: Named chr [1:44811] "alpha-1-B glycoprotein"
 "N-acetyltransferase 2 (arylamine N-acetyltransferase)" "adenosine
 deaminase" "cadherin 2, type 1, N-cadherin (neuronal)" ...
 .. ..- attr(*, "names")= chr [1:44811] "1" "10" "100" "1000" ...
 $ gcounts : 'data.frame': 1362 obs. of 8 variables:
 ..$ GeneID: Factor w/ 1362 levels "100036519","100128285",...: 1346 547 1155 96
 1251 212 638 737 1280 68 ...
 ..$ Symbol: chr [1:1362] "CROCC" "HLA-B" "ZDHHC11" "CASC4" ...
```

```

..$ Name : chr [1:1362] "ciliary rootlet coiled-coil, rootletin" "major
  histocompatibility complex, class I, B" "zinc finger, DHHC-type containing
  11" "cancer susceptibility candidate 4" ...
..$ case_N: int [1:1362] 16 13 8 5 13 4 4 4 4 5 ...
..$ ctrl_N: int [1:1362] 0 4 1 0 6 0 0 0 0 1 ...
..$ case_%: num [1:1362] 1.8 1.462 0.9 0.562 1.462 ...
..$ ctrl_%: num [1:1362] 0 0.349 0.087 0 0.524 0 0 0 0 0.087 ...
..$ Pvalue: num [1:1362] 1.92e-06 6.59e-03 7.44e-03 1.61e-02 2.70e-02 ...
$ support_case: chr [1:726] "9696" "3106" "79844" "113201" ...
$ support_ctrl: chr [1:636] "146857" "55106" "91607" "386757" ...
$ totals :List of 3
..$ all : Named int [1:2] 889 1146
.. ..- attr(*, "names")= chr [1:2] "case" "ctrl"
..$ cnv : Named int [1:2] 692 880
.. ..- attr(*, "names")= chr [1:2] "case" "ctrl"
..$ cnvGen: Named int [1:2] 454 547
.. ..- attr(*, "names")= chr [1:2] "case" "ctrl"
$ coverage : Named chr [1:10] "17060" "4076" "1362" "3164" ...
..- attr(*, "names")= chr [1:10] "Genes in gene-set universe" "Genes hit by CNV
  (before filters)" "Genes hit by CNV (after filters)" "Genes hit by CNV
  (before filters) in gene-sets" ...

```

Its elements are:

- **ann** - Gene annotations (symbols and descriptive names), exactly as in the input (see section 3.3).
- **gcounts** - Data frame containing case/control counts and percentages and the p-value of association for each gene.
- **support_case** - The complete set of “case support genes” (as defined in the description of **extended** earlier in this section) over all gene-sets.
- **support_ctrl** - As above but with “control support genes”.
- **totals** - Counts of case and control samples (all, those with CNVs, and those with genic CNVs).
- **coverage** - Vector of various gene-related statistics; descriptions for each element are in its **names** attribute.

In particular, **gcounts** can be reviewed to see the associations for those top associated genes:

```
> head( geneData(output)$gcounts[ , -3] )
```

	GeneID	Symbol	case_N	ctrl_N	case_%	ctrl_%	Pvalue
9696	9696	CROCC	16	0	1.800	0.000	1.916386e-06
3106	3106	HLA-B	13	4	1.462	0.349	6.591475e-03

79844	79844	ZDHC11	8	1	0.900	0.087	7.443201e-03
113201	113201	CASC4	5	0	0.562	0.000	1.606206e-02
84871	84871	AGBL4	13	6	1.462	0.524	2.696118e-02
147804	147804	LOC147804	4	0	0.450	0.000	3.665167e-02

Comparing this to the 20 gene-sets shown earlier and keeping in mind that the `gcounts` table is sorted by p-value, it appears that CROCC may be a concern. In general, if a gene has a strong association on its own (or if it has a previously known association), it can be removed from the test by including its gene ID in the `rem_genes` parameter in the input (see section 4).

6.2.2 Detailed analysis of gene-set associations

As a further aid in understanding the CNVs and genes contributing to the association results, `enrRes` provides `gstables` – a list of data frames, one for each of the top gene-sets, containing information about the CNVs and corresponding genes affecting the gene-set:

```
> str(enrRes(output)$gstables, max.level=1, list.len=5)
```

List of 200

```
$ GO:0005929:'data.frame':      48 obs. of  13 variables:
.. [list output truncated]
$ GO:0030030:'data.frame':      95 obs. of  13 variables:
.. [list output truncated]
$ GO:0006928:'data.frame':      43 obs. of  13 variables:
.. [list output truncated]
$ GO:0005814:'data.frame':      17 obs. of  13 variables:
.. [list output truncated]
$ GO:0048870:'data.frame':      40 obs. of  13 variables:
.. [list output truncated]
[list output truncated]
```

The structure of the data frame for a particular gene-set is similar to that of `cnvData$cnv` in the input:

```
> options(width=160)
> enrRes(output)$gstables[[2]][10:19,]
```

	CnvID	SampleID	Chr	Coord_i	Coord_f	Type	Length	Gcount	GsGcount	GsID	Class	Genes	Symbols
10	CNV_1222	3266_003	2	110206673	110615080	DEL	408407	2	2	GO:0030030	case	4867	NPHP1
11	CNV_1234	3272_004	21	26100421	26168810	DEL	68389	1	1	GO:0030030	case	351	APP
12	CNV_1378	5007_3	1	144099494	144627859	DEL	528365	17	15	GO:0030030	case	148738	HFE2
13	CNV_1407	5036_4	X	29446046	29557942	DEL	111896	1	1	GO:0030030	case	11141	IL1RAPL1
14	CNV_1435	5065_3	1	17079505	17140083	DEL	60578	1	1	GO:0030030	case	9696	CROCC
15	CNV_1445	5068_3	16	29502984	30127026	DEL	624042	30	21	GO:0030030	case	11151;5595	CORO1A;MAPK3
16	CNV_1449	5072_3	2	50912249	50955087	DEL	42838	1	1	GO:0030030	case	9378	NRXN1
17	CNV_145	13037_463	2	51002576	51157742	DEL	155166	1	1	GO:0030030	case	9378	NRXN1
18	CNV_1455	5081_4	3	1090904	1217096	DEL	126192	1	1	GO:0030030	case	27255	CNTN6
19	CNV_1458	5082_4	17	1754455	1844570	DEL	90115	2	2	GO:0030030	case	146760	RTN4RL1

(Note that the selection above shows only 10 rows from a single data frame in the list.) The extra columns are **Gcount**, showing the total number of genes hit by the CNV (i.e. out of the set of *all* genes – not just the ones that are members of the particular gene-set); **GsGcount**, showing the number of genes hit by the CNV also found amongst *all* gene-sets in the input (i.e. not just those of the particular gene-set); and **Genes** and **Symbols**, finally showing *only* the gene IDs and symbols of those genes hit by the CNV that are members of the particular gene-set.

Examining the data frame for a particular gene-set may reveal that its association due to certain genes may actually be better explained by *other* genes (those that have a clearer functional impact or that have previously been associated with the cases under consideration).

6.2.3 Burden analysis

The enrichment results for a rare CNV/gene-set association test will draw the strongest conclusions when the case and control data are closely matched – i.e. having similar overall CNV and CNV-gene profiles – so that associations arising from the remaining differences can indeed be taken as valid rather than artifacts of the input data. The “burden” statistics in `burdenGs` and `burdenSample`, described below, are provided for this purpose.

Taking a look thus at the `burdenSample` statistics:

```
> burdenSample(output)

$SamplesCNV
$SamplesCNV$summary
$SamplesCNV$summary$case
  LogLenMean LogLenTot  CNV_N  GenCNV_N  GsGenCNV_N  Gene_N_Mean  GsGene_N_Mean  Gene_N_Tot  GsGene_N_Tot
Min    4.481772  4.481772  1.000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
Q1     4.716148  4.849929  1.000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
Mean   4.952673  5.147046  1.776012  0.9089595  0.7947977  1.025864  0.7986237  1.884393  1.465318
Median 4.890383  5.126139  1.500000  1.0000000  1.0000000  0.500000  0.5000000  1.000000  1.000000
Q3     5.113215  5.410046  2.000000  1.0000000  1.0000000  1.000000  1.0000000  2.000000  2.000000
Max    6.268872  6.569902  7.000000  6.0000000  5.0000000  31.000000  21.0000000  31.000000  24.000000

$SamplesCNV$summary$ctrl
  LogLenMean LogLenTot  CNV_N  GenCNV_N  GsGenCNV_N  Gene_N_Mean  GsGene_N_Mean  Gene_N_Tot  GsGene_N_Tot
Min    4.477136  4.477136  1.000000  0.0000000  0.0000000  0.000000  0.0000000  0.000000  0.000000
Q1     4.742337  4.852478  1.000000  0.0000000  0.0000000  0.000000  0.0000000  0.000000  0.000000
Mean   4.978547  5.164490  1.735227  0.8147727  0.6886364  0.916875  0.6614962  1.582955  1.163636
Median 4.911761  5.144712  1.000000  1.0000000  1.0000000  0.500000  0.3333333  1.000000  1.000000
Q3     5.163548  5.415993  2.000000  1.0000000  1.0000000  1.000000  1.0000000  2.000000  2.000000
Max    6.466677  6.466677  6.000000  5.0000000  4.0000000  19.000000  9.5000000  30.000000  21.000000

$SamplesCNV$pvalue
  LogLenMean LogLenTot  CNV_N  GenCNV_N  GsGenCNV_N  Gene_N_Mean  GsGene_N_Mean  Gene_N_Tot  GsGene_N_Tot
case > ctrl 0.93867665 0.8005281 0.201231 0.01331392 0.003815777 0.1063544 0.01592546 0.02198636 0.004259239
case < ctrl 0.06132335 0.1994719 0.798769 0.98668608 0.996184223 0.8936456 0.98407454 0.97801364 0.995740761

$SamplesCNV$no_cnv_proportion
```

```

$SamplesCNV$no_cnv_proportion$PropEstimates
      case      ctrl
0.7784027 0.7678883

```

```

$SamplesCNV$no_cnv_proportion$Pvalue
[1] 0.6115485

```

The first two tables, `$SamplesCNV$summary$case` and `$SamplesCNV$summary$ctrl`, show summary statistics across individual case and control samples. `LogLenMean` and `LogLenTot` are the (base 10) logarithm of the mean and total lengths of the CNVs in a sample; `CNV_N`, `GenCNV_N`, and `GsGenCNV_N` are the number of all CNVs, genic CNVs, and gene-set genic CNVs in the sample; and finally `Gene_N_Mean`, `GsGene_N_Mean`, `Gene_N_Tot`, and `GsGene_N_Tot` are the mean and total counts of genes and genic genes *per CNV* in the sample. Comparing these two tables shows that the case and control data sets are relatively similar in the example above.

The next table, `$SamplesCNV$pvalue`, shows the results of a t-test done for each of the statistics above comparing cases to controls. If any of these p-values is significant, gene-sets could be systematically inflated. `$SamplesCNV$no_cnv_proportion` likewise shows the fraction of samples in cases and controls that *do* contain the particular CNV type set in the test parameters (i.e. usually one of "DUP" or "DEL" – see section 4) and the results of a t-test comparison between them – with similar implications if the p-value there is significant.

Taking a look now at the `burdenGs` statistics:

```
> burdenGs(output)
```

```
$coverage
```

	All	case	ctrl
Sample N in the study, no filters	2035.0000000	889.0000000	1146.0000000
Sample N with at least one cnv, no filters	2035.0000000	889.0000000	1146.0000000
Sample N with at least one cnv	1572.0000000	692.0000000	880.0000000
Sample % with at least one cnv (on tot)	0.7724816	0.7784027	0.7678883
Sample N with at least one genic cnv	1001.0000000	454.0000000	547.0000000
Sample % with at least one genic cnv (on tot)	0.4918919	0.5106862	0.4773124
Sample N with at least one perturbed gene-set	892.0000000	412.0000000	480.0000000
Sample % with at least one perturbed gene-set (on tot)	0.4383292	0.4634421	0.4188482
Gene-set N with at least one sample	3369.0000000	3059.0000000	2798.0000000
Gene-set % with at least one sample	0.9051585	0.8218700	0.7517464

```
$pairs
```

	All	case	ctrl
N of sample-gs pair, >= 1 CNV-perturbed gene	3.777900e+04	2.031300e+04	1.746600e+04
% of sample-gs pair, >= 1 CNV-perturbed gene (on all pairs)	4.987807e-03	6.138976e-03	4.094798e-03
N of sample-gs pair, >= 2 CNV-perturbed gene	2.335000e+03	1.180000e+03	1.155000e+03
% of sample-gs pair, >= 2 CNV-perturbed gene (on positive pairs)	6.180682e-02	5.809088e-02	6.612848e-02
N of sample-gs pair, >= 3 CNV-perturbed gene	5.230000e+02	2.810000e+02	2.420000e+02
% of sample-gs pair, >= 3 CNV-perturbed gene (on positive pairs)	1.384367e-02	1.383351e-02	1.385549e-02
N of sample-gs pair, >= 2 CNV	3.050000e+02	1.550000e+02	1.500000e+02
% of sample-gs pair, >= 2 CNV (on positive pairs)	8.073268e-03	7.630581e-03	8.588114e-03
N of sample-gs pair, >= 2 CNV on distinct chr.	2.370000e+02	1.320000e+02	1.050000e+02
% of sample-gs pair, >= 2 CNV on distinct chr. (on positive pairs)	6.273326e-03	6.498302e-03	6.011680e-03

The first table shows basic statistics for all, case, and control samples:

- Total number of samples;
- Number of samples with at least one *pre*-filtered CNV;
- Number and percentage of samples with at least one *post*-filtered CNV (“(on tot)” simply indicates that the percentage is taken on the total number of the particular type of sample);
- Number and percentage of samples with at least one genic CNV;
- Number and percentage of samples with at least one CNV hitting a gene-set under consideration; and finally
- Number and percentage of gene-sets hit by at least one sample.

As with `burdenSample`, the values in this first table show that the case and control data sets in this example are appropriately matched.

The second table above shows statistics for all, case, and control (sample, gene-set) *pairs* and requires some explanation. A “(sample, gene-set) pair” in the context of these statistics is a cell in the initial matrix of perturbation counts formed by tabulating all gene-sets against all samples, where the perturbation count is the number of genes in the gene-set that are hit by CNVs in the sample. Nonzero values in this initial matrix are then truncated into a “perturbation score” (i.e. values greater than 1 are set to 1); this matrix of perturbation scores is the one, as described in the Overview section, that is in turn used to compute the contingency tables used in the Fisher Exact Test for each gene-set. The rows in the second table above are thus taken from the *initial* matrix – as follows:

- Number and percentage of (sample, gene-set) pairs having at least one gene of interest hit by CNVs in the sample. “(on all pairs)” indicates that the percentage is taken out of all cells in the matrix.
- Number and percentage of (sample, gene-set) pairs having at least 2 / at least 3 genes of interest hit by CNVs in the sample. “(on positive pairs)” indicates that the percentage is taken out of all *nonzero* cells in the matrix.
- Number and percentage of (sample, gene-set) pairs having at least 2 CNVs in the sample.
- Number and percentage of (sample, gene-set) pairs having at least 2 CNVs in the sample – where the CNVs are distinct chromosomes.

The rationale for these statistics is twofold: on the one hand, it is another check to ensure that the case and control data sets are well-matched; on the other hand, if it turns out that a substantial number of CNVs are hitting more than one gene-set, it may be an indication to apply a more sophisticated association test (such as the trend test). In the `burdenGs` output above, the statistics again show that the cases and controls are well-matched, and the percentage of CNVs hitting more than one gene-set is evidently low (around 6%). It should be noted that the size of the gene-sets will affect these statistics (larger gene-sets will increase the likelihood that the CNVs are hitting genes from more than one gene-set).

7 References

- [1] Pinto, D et al. Functional impact of global rare copy number variation in autism spectrum disorders. *Nature*. 2010 Jul 15; **466**(7304): 368–72.
- [2] A. C. Lionel et al. Rare Copy Number Variation Discovery and Cross-Disorder Comparisons Identify Risk Genes for ADHD. *Sci. Transl. Med.* **3**, 95ra75 (2011).
- [3] C. R. Marshall et al. Structural Variation of Chromosomes in Autism Spectrum Disorder. *Am J Hum Genet.* 2008 Feb 8; **82**(2): 477–488.