Package 'cnvGSA'

October 14, 2021

Type Package

Title Gene Set Analysis of (Rare) Copy Number Variants

Version 1.36.0

Date 2015-03-02

Author Daniele Merico <daniele.merico@gmail.com>, Robert Ziman <rziman@gmail.com>; packaged by Joseph Lugo <joseph.r.lugo@gmail.com>

Maintainer Joseph Lugo <joseph.r.lugo@gmail.com>

Description
This package is intended to facilitate gene-set association with rare CNVs in case-control studies.

License LGPL

LazyLoad yes

Depends brglm, doParallel, foreach, GenomicRanges, methods, splitstackshape

Suggests cnvGSAdata, org.Hs.eg.db

biocViews MultipleComparison

git_url https://git.bioconductor.org/packages/cnvGSA

git_branch RELEASE_3_13

git_last_commit acbd086

git_last_commit_date 2021-05-19

Date/Publication 2021-10-14

R topics documented:

`cnvGSA-package` *Gene-set Analysis of (Rare) Copy Number Variants*

Description

cnvGSA is an R package for testing the rare gene-set variant burden in case-control studies of copy number variation (CNV).

Details

In cnvGSA, subjects are treated as statistical sampling units. Subject-level covariates that may act as confounders can be provided by the user (e.g. sex, ethnicity, CNV genotyping platform, CNV genotyping site, array quality metrics, etc.). The gene-set burden is tested using a logistic regression approach. Two logistic regression models are fit: model A includes the subject-level covariates and a variable quantifying global CNV burden for each subject (total CNV length, or total number of CNV-overlapped genes per subject, etc.); model B includes all variables present in model A, plus the number of CNV-overlapped genes that are members of the gene-set being tested. Presence of significantly higher burden in cases compared to controls for the gene-set of interest is then tested by comparing the two models using a deviance chi-square test, as implemented by anova.glm.

Author(s)

Daniele Merico <daniele.merico@gmail.com>, Robert Ziman <rziman@gmail.com>; packaged by Joseph Lugo <joseph.r.lugo@gmail.com>

`cnvGSAgsTables` *Creates the gene-set tables for each gene-set.*

Description

Creates the gene-set tables for each gene-set.

Usage

```
cnvGSAgsTables(cnvGSA.in, cnvGSA.out)
```

Arguments

`cnvGSA.in`	A CnvGSAInput S4 object.
`cnvGSA.out`	A CnvGSAOutput S4 object.

Value

A list where each object is a table corresponding to one gene-set.

Examples

```
library(cnvGSAdata)
data(cnvGSA_output_example)
## See vignette for full details and worked example
```

`cnvGSAIn` *Creating the input S4 object needed to run the script.*

Description

Creating the input S4 object needed to run the script.

Usage

```
cnvGSAIn(configFile, cnvGSA.in)
```

Arguments

`configFile`	The file name for the config file including the full path.
`cnvGSA.in`	A CnvGSAInput S4 object.

Value

A cnvGSAInput S4 object.

Examples

```
library(cnvGSAdata)
data(cnvGSA_input_example)
## See vignette for full details and worked example
```

`CnvGSAInput-class` *Class* `"CnvGSAInput"`

Description

Container class for the input structures required by the main function (i.e. cnvGSALogRegTest()).

Slots

`config.ls`**:** Object of class `"list"` containing the file names and paths.

`params.ls`**:** Object of class `"list"` containing the test parameters.

`cnvData.ls`**:** Object of class `"list"` containing CNV data.

`phData.ls`**:** Object of class `"list"` containing phenotype and covariate data.

`gsData.ls`**:** Object of class `"list"` containing gene-set data.

`geneID.ls`**:** Object of class `"list"` containing Gene ID data.

Constructor

`CnvGSAInput( config.ls,params.ls,cnvData.ls,phData.ls,gsData.ls,geneID.ls)`: Creates a CnvGSAInput object.

`config.ls` Structure containing the file names and paths.

`params.ls` Structure containing main test parameters.

`cnvData.ls` Structure containing CNV data along with sample-to-class information and filters.

`phData.ls` Structure containing phenotype and covariate data.

`gsData.ls` Structure containing gene-set data.

`geneID.ls` Structure containing Gene ID data.

See the vignette for complete details on each of these structures as well as a full example of how to load them.

Methods

config.ls `signature(obj = "CnvGSAInput")`: Gets `config.ls`.

config.ls<- `signature(obj = "CnvGSAInput")`: Sets `config.ls`.

params.ls `signature(obj = "CnvGSAInput")`: Gets `params.ls`.

params.ls<- `signature(obj = "CnvGSAInput")`: Sets `params.ls`.

cnvData.ls `signature(obj = "CnvGSAInput")`: Gets `cnvData.ls`.

cnvData.ls<- `signature(obj = "CnvGSAInput")`: Sets `cnvData.ls`.

phData.ls `signature(obj = "CnvGSAInput")`: Gets `phData.ls`.

phData.ls<- `signature(obj = "CnvGSAInput")`: Sets `phData.ls`.

gsData.ls `signature(obj = "CnvGSAInput")`: Gets `gsData.ls`.

gsData.ls<- `signature(obj = "CnvGSAInput")`: Sets `gsData.ls`.

geneID.ls `signature(obj = "CnvGSAInput")`: Gets `geneID.ls`.

geneID.ls<- `signature(obj = "CnvGSAInput")`: Sets `geneID.ls`.

Author(s)

Joseph Lugo <joseph.r.lugo@gmail.com>

Examples

```
## See vignette for full details and worked example
```

cnvGSAlogRegTest *Performing the logistic regression tests on the CNV data.*

Description

This test uses 4 different correction models and requires a case control study. It looks at odds ratios and calculates statistics for the gene-set collection.

Usage

```
cnvGSAlogRegTest(cnvGSA.in, cnvGSA.out)
```

Arguments

cnvGSA.in	A CnvGSAInput S4 object.
cnvGSA.out	A CnvGSAOutput S4 object.

Value

A list of one or two objects depending on whether or not the user includes the known loci in the analysis. Each object in the list contains the regression results for each gene set.

Examples

```
library(cnvGSAdata)
data(cnvGSA_output_example)
## See vignette for full details and worked example
```

CnvGSAOutput-class *Class* "CnvGSAOutput"

Description

Container class for the output structures produced by the main function (i.e. cnvGSAlogRegTest()).

Slots

phData.ls**:** Object of class "list" containing phenotype and covariate data.

res.ls**:** Object of class "list" containing burden analysis results for gene-sets.

gsTables.ls**:** Object of class "list" containing the gene-set tables.

gsData.ls**:** Object of class "list" containing gene-set data.

Methods

gsTables.ls `signature(obj = "CnvGSAOutput")`: Gets `gsTables.ls`.

res.ls `signature(obj = "CnvGSAOutput")`: Gets `res.ls`.

phData.ls `signature(obj = "CnvGSAOutput")`: Gets `phData.ls`.

gsData.ls `signature(obj = "CnvGSAOutput")`: Gets `gsData.ls`.

Author(s)

Joseph Lugo <joseph.r.lugo@gmail.com>

Examples

```
## See vignette for full details and worked example
```

---

f.enrFiles *Prepares the files for the enrichment maps.*

Description

Prepares the files for the enrichment maps.

Usage

```
f.enrFiles(cnvGSA.in, cnvGSA.out)
```

Arguments

`cnvGSA.in`	A CnvGSAInput S4 object.
`cnvGSA.out`	A CnvGSAOutput S4 object.

Value

Returns a list with the data frames of the GMT file and the generic file.

Examples

```
## See vignette for full details and worked example
```

f.makeViz — *Creates the plots from the CnvGSAOutput data.*

Description

Creates the plots from the CnvGSAOutput data.

Usage

```
f.makeViz(cnvGSA.in, cnvGSA.out)
```

Arguments

cnvGSA.in	A CnvGSAInput S4 object.
cnvGSA.out	A CnvGSAOutput S4 object.

Value

Creates the plots to better understand the output.

Examples

```
## See vignette for full details and worked example
```

f.readConfig — *Reading in the config file.*

Description

This function is used to read in all the values from the config file and change them in the S4 objects used throughout the scripts. If you would like to reload the config values, you will want to run this function.

Usage

```
f.readConfig(configFile, cnvGSA.in)
```

Arguments

configFile	The file name and full path for the config file.
cnvGSA.in	A CnvGSAInput S4 object.

Value

A CnvGSAInput object with the updated config.ls and params.ls objects.

Examples

```
library(cnvGSAdata)
data(cnvGSA_input_example)
## See vignette for full details and worked example
```

Index