Package ‘BLMA’

April 25, 2017

Date 2017-01-16
Type Package
Title BLMA: A package for bi-level meta-analysis
Version 1.0.0
Author Tin Nguyen <tin@wayne.edu> and Sorin Draghici <sorin@wayne.edu>
Maintainer Tin Nguyen <tin@wayne.edu>
Description Suit of tools for bi-level meta-analysis. The package can be used in a wide range of applications, including general hypothesis testings, differential expression analysis, functional analysis, and pathway analysis.
biocViews GeneSetEnrichment, Pathways, DifferentialExpression, Microarray
License GPL (>=2)
Depends ROntoTools, GSA, PADOG, limma, graph, stats, utils, parallel, Biobase
Suggests RUnit, BiocGenerics
RoxygenNote 5.0.1
NeedsCompilation no

R topics documented:

addCLT .......................... 2
bilevelAnalysisClassic ................. 3
bilevelAnalysisGene ..................... 4
bilevelAnalysisGeneset ................. 5
bilevelAnalysisPathway .................. 8
fisherMethod ........................ 10
GSE17054 .......................... 11
GSE33223 .......................... 11
GSE42140 .......................... 12
GSE57194 .......................... 12
intraAnalysisClassic ................. 13
intraAnalysisGene ..................... 14
loadKEGGPathways ..................... 16
stoufferMethod ....................... 16

Index 18
addCLT

The additive method for meta-analysis

Description

Combine independent studies using the average of p-values

Usage

addCLT(x)

Arguments

x is an array of independent p-values

Details

This method is based on the fact that sum of independent uniform variables follow the Irwin-Hall distribution [1a,1b]. When the number of p-values is small (n<20), the distribution of the average of p-values can be calculated using a linear transformation of the Irwin-Hall distribution. When n is large, the distribution is approximated using the Central Limit Theorem to avoid underflow/overflow problems [2,3,4,5].

Value

combined p-value

Author(s)

Tin Nguyen and Sorin Draghici

References

[1a] P. Hall. The distribution of means for samples of size n drawn from a population in which the variate takes values between 0 and 1, all such values being equally probable. Biometrika, 19(3-4):240-244, 1927.
bilevelAnalysisClassic

Bi-level meta-analysis in conjunction with a classical hypothesis testing method

Description

Perform a bi-level meta-analysis in conjunction with any of the classical hypothesis testing methods, such as t-test, Wilcoxon test, etc.

Usage

bilevelAnalysisClassic(x, y = NULL, splitSize = 5, metaMethod = addCLT, func = t.test, p.value = "p.value", ...)

Arguments

- x: a list of numeric vectors
- y: an optional list of numeric vectors
- splitSize: the minimum number of size in each split sample. splitSize should be at least 3. By default, splitSize=5
- metaMethod: the method used to combine p-values. This should be one of addCLT (additive method [1]), fishersMethod (Fisher’s method [5]), stoufferMethod (Stouffer’s method [6]), max (maxP method [7]), or min (minP method [8])
- func: the name of the hypothesis test. By default func=t.test
- p.value: the component that returns the p-value after performing the test provided by the func parameter. For example, the function t-test returns the class "htest" where the component "p.value" is the p-value of the test. By default, p.value="p.value"
- ...: additional parameters for func

Details

This function performs a bi-level meta-analysis for the lists of samples [1]. It performs intra-experiment analyses to compare the vectors in x against the corresponding vectors in y using the function intraAnalysisClassic in conjunction with the test provided in func. For example, it compares the first vector in x with the first vector in y, the second vector in x with the second vector in y, etc. When y is null, then the comparisons are reduced to one-sample tests. After these comparisons, we have a list of p-values, one for each comparison. The function then combines these p-values to obtain a single p-value using metaMethod.
Value

the combined p-value

Author(s)

Tin Nguyen and Sorin Draghici

References


See Also

intraAnalysisClassic, intraAnalysisGene, bilevelAnalysisGene

Examples

set.seed(1)
l1 <- lapply(as.list(seq(3)), FUN=function (x) rnorm(n=10, mean=1))
l1
# one-sample t-test
lapply(l1, FUN=function(x) t.test(x, alternative="greater")$p.value)
# combining the p-values of one-sample t-tests:
addCLT(unlist(lapply(l1, FUN=function(x) t.test(x, alter="g")$p.value)))
# Bi-level meta-analysis
bilevelAnalysisClassic(x=l1, alternative="greater")

bilevelAnalysisGene  Bi-level meta-analysis of multiple expression datasets at the gene-level

Description

Perform a bi-level meta-analysis in conjunction with the moderate t-test (limma package) for the purpose of differential expression analysis of multiple gene expression datasets

Usage

bilevelAnalysisGene(dataList, groupList, splitSize = 5, metaMethod = addCLT)

Arguments

dataList  a list of datasets. Each dataset is a data frame where the rows are the gene IDs and the columns are the samples

groupList a list of vectors. Each vector represents the phenotypes of the corresponding dataset in dataList, which are either 'c' (control) or 'd' (disease).

splitSize  the minimum number of disease samples in each split dataset. splitSize should be at least 3. By default, splitSize=5

metaMethod the method used to combine p-values. This should be one of addCLT (additive method [1]), fishersMethod (Fisher’s method [5]), stoufferMethod (Stouffer’s method [6]), max (maxP method [7]), or min (minP method [8])
bilevelAnalysisGeneset

Details
The bi-level framework combines the datasets at two levels: an intra-experiment analysis, and an inter-experiment analysis [1]. At the intra-experiment analysis, the framework splits a dataset into smaller datasets, performs a moderated t-test (limma package) on split datasets, and then combines p-values of individual genes using `metaMethod`. At the inter-experiment analysis, the p-values obtained for each individual datasets are combined using `metaMethod`.

Value
A data frame containing the following components:

- **rownames**: gene IDs that are common in all datasets
- **pLimma**: the p-values obtained by combining pLimma values of individual datasets
- **pLimma.fdr**: FDR-corrected p-values of pLimma
- **pBilevel**: the p-values obtained from combining pIntra values of individual datasets
- **pBilevel.fdr**: FDR-corrected p-values of pBilevel

Author(s)
Tin Nguyen and Sorin Draghici

References

See Also
bilevelAnalysisGene, intraAnalysisClassic

Examples
```r
dataSets <- c("GSE17054", "GSE57194", "GSE33223", "GSE42140")
data(list=dataSets, package="BLMA")
names(dataSets) <- dataSets
dataList <- lapply(dataSets, function(dataset) get(paste0("data_", dataset)))
groupList <- lapply(dataSets, function(dataset) get(paste0("group_", dataset)))
Z <- bilevelAnalysisGene(dataList = dataList, groupList = groupList)
head(Z)
```

bilevelAnalysisGeneset

Bi-level meta-analysis – applied to geneset enrichment analysis

Description
Perform a bi-level meta-analysis in conjunction with geneset enrichment methods (ORA/GSA/PADOG) to integrate multiple gene expression datasets.
bilevelAnalysisGeneset

Usage

bilevelAnalysisGeneset(gslist, gs.names, dataList, groupList, splitSize = 5, metaMethod = addCLT, enrichment = "ORA", pCutoff = 0.05, percent = 0.05, mc.cores = 1, ...)

Arguments

- **gslist**: a list of gene sets.
- **gs.names**: names of the gene sets.
- **dataList**: a list of datasets to be combined. Each dataset is a data frame where the rows are the gene IDs and the columns are the samples.
- **groupList**: a list of vectors. Each vector represents the phenotypes of the corresponding dataset in dataList. The elements of each vector are either 'c' (control) or 'd' (disease).
- **splitSize**: the minimum number of disease samples in each split dataset. splitSize should be at least 3. By default, splitSize=5
- **metaMethod**: the method used to combine p-values. This should be one of addCLT (additive method [1]), fisherMethod (Fisher’s method [5]), stoufferMethod (Stouffer’s method [6]), max (maxP method [7]), or min (minP method [8])
- **enrichment**: the method used for enrichment analysis. This should be one of "ORA", "GSA", or "PADOG". By default, enrichment is set to "ORA".
- **pCutoff**: cutoff p-value used to identify differentially expressed (DE) genes. This parameter is used only when the enrichment method is "ORA". By default, pCutoff=0.05 (five percent)
- **percent**: percentage of genes with highest foldchange to be considered as differentially expressed (DE). This parameter is used when the enrichment method is "ORA". By default, percent=0.05 (five percent). Please note that only genes with p-value less than pCutoff will be considered
- **mc.cores**: the number of cores to be used in parallel computing. By default, mc.cores=1
- **...**: additional parameters of the GSA/PADOG functions

Details

The bi-level framework combines the datasets at two levels: an intra-experiment analysis, and an inter-experiment analysis [1]. At the intra-level analysis, the framework splits a dataset into smaller datasets, performs enrichment analysis for each split dataset (using ORA [2], GSA [3], or PADOG [4]), and then combines the results of these split datasets using metaMethod. At the inter-level analysis, the results obtained for individual datasets are combined using metaMethod

Value

A data frame (rownames are geneset/pathway IDs) that consists of the following information:

- **Name**: name/description of the corresponding pathway_geneset
- Columns that include the pvalues obtained from the intra-experiment analysis of individual datasets
- **pBLMA**: p-value obtained from the inter-experiment analysis using addCLT
- **rBLMA**: ranking of the geneset/pathway using addCLT
- **pBLMA.fdr**: FDR-corrected p-values
**Author(s)**

Tin Nguyen and Sorin Draghici

**References**


**See Also**

bilevelAnalysisPathway, phyper, GSA, padog

**Examples**

```r
# load KEGG pathways and create gene sets
x <- loadKEGGPathways()
gslist <- lapply(x$kpg,FUN=function(y){return (nodes(y));})
gs.names <- x$kpn[names(gslist)]

# load example data
dataSets <- c("GSE17054", "GSE57194", "GSE33223", "GSE42140")
data(list=dataSets, package="BLMA")

names(dataSets) <- dataSets
datalist <- lapply(dataSets, function(dataset) get(paste0("data_", dataset)))
groupList <- lapply(dataSets, function(dataset) get(paste0("group_", dataset)))

# perform bi-level meta-analysis in conjunction with ORA
ORAComb <- bilevelAnalysisGeneset(gslist, gs.names, datalist, groupList, enrichment = "ORA")
head(ORAComb[, c("Name", "pBLMA", "pBLMA.fdr", "rBLMA")])

# perform bi-level meta-analysis in conjunction with GSA
GSAComb <- bilevelAnalysisGeneset(gslist, gs.names, datalist, groupList, enrichment = "GSA", nperms = 200, random.seed = 1)
head(GSAComb[, c("Name", "pBLMA", "pBLMA.fdr", "rBLMA")])

# perform bi-level meta-analysis in conjunction with PADOG
set.seed(1)
PADOGComb <- bilevelAnalysisGeneset(gslist, gs.names, datalist, groupList, enrichment = "PADOG", NI=200)
head(PADOGComb[, c("Name", "pBLMA", "pBLMA.fdr", "rBLMA")])
```

# load KEGG pathways and create gene sets
x <- loadKEGGPathways()
gslist <- lapply(x$kpg,FUN=function(y){return (nodes(y));})
gs.names <- x$kpn[names(gslist)]

# load example data
dataSets <- c("GSE17054", "GSE57194", "GSE33223", "GSE42140")
data(list=dataSets, package="BLMA")

names(dataSets) <- dataSets
datalist <- lapply(dataSets, function(dataset) get(paste0("data_", dataset)))
groupList <- lapply(dataSets, function(dataset) get(paste0("group_", dataset)))

# perform bi-level meta-analysis in conjunction with ORA
ORAComb <- bilevelAnalysisGeneset(gslist, gs.names, datalist, groupList, enrichment = "ORA")
head(ORAComb[, c("Name", "pBLMA", "pBLMA.fdr", "rBLMA")])

# perform bi-level meta-analysis in conjunction with GSA
GSAComb <- bilevelAnalysisGeneset(gslist, gs.names, datalist, groupList, enrichment = "GSA", nperms = 200, random.seed = 1)
head(GSAComb[, c("Name", "pBLMA", "pBLMA.fdr", "rBLMA")])

# perform bi-level meta-analysis in conjunction with PADOG
set.seed(1)
PADOGComb <- bilevelAnalysisGeneset(gslist, gs.names, datalist, groupList, enrichment = "PADOG", NI=200)
head(PADOGComb[, c("Name", "pBLMA", "pBLMA.fdr", "rBLMA")])
```
bilevelAnalysisPathway

Bi-level meta-analysis – applied to pathway analysis

Description

Perform a bi-level meta-analysis conjunction with Impact Analysis to integrate multiple gene expression datasets

Usage

bilevelAnalysisPathway(kpg, kpn, dataList, groupList, splitSize = 5, metaMethod = addCLT, pCutoff = 0.05, percent = 0.05, mc.cores = 1, nboot = 200, seed = 1)

Arguments

kpg  list of pathway graphs as objects of type graph (e.g., graphNEL)
kpn  names of the pathways.
dataList  a list of datasets to be combined. Each dataset is a data frame where the rows are the gene IDs and the columns are the samples.
groupList  a list of vectors. Each vector represents the phenotypes of the corresponding dataset in dataList, which are either ‘c’ (control) or ‘d’ (disease).
splitSize  the minimum number of disease samples in each split dataset. splitSize should be at least 3. By default, splitSize=5
metaMethod  the method used to combine p-values. This should be one of addCLT (additive method [1]), fisherMethod (Fisher’s method [5]), stoufferMethod (Stouffer’s method [6]), max (maxP method [7]), or min (minP method [8])
pCutoff  cutoff p-value used to identify differentially expressed (DE) genes. This parameter is used only when the enrichment method is "ORA". By default, pCutoff=0.05 (five percent)
percent  percentage of genes with highest foldchange to be considered as differentially expressed (DE). This parameter is used when the enrichment method is "ORA". By default percent=0.05 (five percent). Please note that only genes with p-value less than pCutoff will be considered
mc.cores  the number of cores to be used in parallel computing. By default, mc.cores=1
nboot  number of bootstrap iterations. By default, nboot=200
seed  seed. By default, seed=1.

Details

The bi-level framework combines the datasets at two levels: an intra-experiment analysis, and an inter-experiment analysis [1]. At the intra-level analysis, the framework splits a dataset into smaller datasets, performs pathway analysis for each split dataset using Impact Analysis [2,3], and then combines the results of these split datasets using metaMethod. At the inter-level analysis, the results obtained for individual datasets are combined using metaMethod.
Value

A data frame (rownames are geneset/pathway IDs) that consists of the following information:

- **Name**: name/description of the corresponding pathway/geneset
- Columns that include the p-values obtained from the intra-experiment analysis of individual datasets
- **pBLMA**: p-value obtained from the inter-experiment analysis using addCLT
- **rBLMA**: ranking of the geneset/pathway using addCLT
- **pBLMA.fdr**: FDR-corrected p-values

Author(s)

Tin Nguyen and Sorin Draghici

References


See Also

*bilevelAnalysisGeneset*, *pe*, *phyper*

Examples

```r
# load KEGG pathways
dx <- loadKEGGPathways()

# load example data
dataSets <- c("GSE17054", "GSE57194", "GSE33223", "GSE42140")
data(list=dataSets, package="BLMA")

names(dataSets) <- dataSets
dataList <- lapply(dataSets, function(dataset) get(paste0("data_", dataset)))
groupList <- lapply(dataSets, function(dataset) get(paste0("group_", dataset)))

IAComb <- bilevelAnalysisPathway(x$kpg, x$kpn, dataList, groupList)
head(IAComb[, c("Name", "pBLMA", "pBLMA.fdr", "rBLMA")])
```
fisherMethod  

Fisher’s method for meta-analysis

Description

Combine independent p-values using the minus log product

Usage

fisherMethod(x)

Arguments

x is an array of independent p-values

Details

Considering a set of \( m \) independent significance tests, the resulted p-values are independent and uniformly distributed between 0 and 1 under the null hypothesis. Fisher’s method uses the minus log product of the p-values as the summary statistic, which follows a chi-square distribution with \( 2m \) degrees of freedom. This chi-square distribution is used to calculate the combined p-value.

Value

combined p-value

Author(s)

Tin Nguyen and Sorin Draghici

References


See Also

stoufferMethod, addCLT

Examples

```r
x <- rep(0, 10)
fisherMethod(x)
```
```r
x <- runif(10)
fisherMethod(x)
```
GSE17054  

**Gene expression dataset GSE17054 from Majeti et al.**

**Description**

This dataset consists of 5 acute myeloid leukemia and 4 control samples. The data frame data_GSE17054 includes the expression data while the vector group_GSE17054 includes the grouping information.

**Usage**

```r
data(GSE17054)
```

**Format**

- `data_GSE17054` is a data frame with 4738 rows and 9 columns. The rows represent the genes and the columns represent the samples.
- `group_GSE17054` is a vector that represents the sample grouping for `data_GSE17054`. The elements of `group_GSE17054` are either 'c' (control) or 'd' (disease).

**Source**


**References**


---

GSE33223  

**Gene expression dataset GSE33223 from Bacher et al.**

**Description**

This dataset consists of 20 acute myeloid leukemia and 10 control samples. The data frame data_GSE33223 includes the expression data while the vector group_GSE33223 includes the grouping information.

**Usage**

```r
data(GSE33223)
```

**Format**

- `data_GSE33223` is a data frame with 4114 rows and 30 columns. The rows represent the genes and the columns represent the samples.
- `group_GSE33223` is a vector that represents the sample grouping for `data_GSE33223`. The elements of `group_GSE33223` are either 'c' (control) or 'd' (disease).

**Source**

References


<table>
<thead>
<tr>
<th>GSE42140</th>
<th>The gene expression dataset GSE42140 obtained from Gene Expression Omnibus</th>
</tr>
</thead>
</table>

Description

This dataset consists of 26 acute myeloid leukemia and 5 control samples. The data frame data_GSE42140 includes the expression data while the vector group_GSE42140 includes the grouping information.

Usage

data(GSE42140)

Format

data_GSE42140 is a data frame with 4114 rows and 31 columns. The rows represent the genes and the columns represent the samples.

group_GSE42140 is a vector that represents the sample grouping for data_GSE42140. The elements of group_GSE42140 are either 'c' (control) or 'd' (disease).

References


<table>
<thead>
<tr>
<th>GSE57194</th>
<th>Gene expression dataset GSE57194 from Abdul-Nabi et al.</th>
</tr>
</thead>
</table>

Description

This dataset consists of 6 acute myeloid leukemia and 6 control samples. The data frame data_GSE57194 includes the expression data while the vector group_GSE57194 includes the grouping information.

Usage

data(GSE57194)

Format

data_GSE57194 is a data frame with 4114 rows and 12 columns. The rows represent the genes and the columns represent the samples.

group_GSE57194 is a vector that represents the sample grouping for data_GSE57194. The elements of group_GSE57194 are either 'c' (control) or 'd' (disease).
Source


References

Abdul-Nabi et al. In vitro transformation of primary human CD34+ cells by AML fusion onco-

intraAnalysisClassic

Intra-experiment analysis in conjunction with classical hypothesis tests

Description

Perform an intra-experiment analysis in conjunction with any of the classical hypothesis testing methods, such as t-test, Wilcoxon test, etc.

Usage

intraAnalysisClassic(x, y = NULL, splitSize = 5, metaMethod = addCLT, func = t.test, p.value = "p.value", ...)

Arguments

x
  a numeric vector of data values

y
  an optional numeric vector of values

splitSize
  the minimum number of size in each split sample. splitSize should be at least 3. By default, splitSize=5

metaMethod
  the method used to combine p-values. This should be one of addCLT (additive method \[1\]), fishersMethod (Fisher’s method \[5\]), stoufferMethod (Stouffer’s method \[6\]), max (maxP method \[7\]), or min (minP method \[8\])

func
  the name of the hypothesis test. By default func=t.test

p.value
  the component that returns the p-value after performing the test provided by the func parameter. For example, the function t-test returns the class "htest" where the component "p.value" is the p-value of the test. By default, p.value="p.value"

... additional parameters for func

Details

This function performs an intra-experiment analysis for the given sample(s) \[1\]. Given x as the numeric vector, this function first splits x into smaller samples with size splitSize, performs hypothesis testing using func, and then combines the p-values using metaMethod

Value

intra-experiment p-value
intraAnalysisGene

Author(s)

Tin Nguyen and Sorin Draghici

References


See Also

bilevelAnalysisClassic, intraAnalysisGene, bilevelAnalysisGene

Examples

set.seed(1)
x <- rnorm(10, mean = 0)
# p-value obtained from a one-sample t-test
t.test(x, mu=1, alternative = "less")$p.value
# p-value obtained from an intra-experiment analysis
intraAnalysisClassic(x, func=t.test, mu=1, alternative = "less")

# p-value obtained from a one-sample wilcoxon test
wilcox.test(x, mu=1, alternative = "less")$p.value
# p-value obtained from an intra-experiment analysis
intraAnalysisClassic(x, func= wilcox.test, mu=1, alternative = "less")

set.seed(1)
x <- rnorm(20, mean=0); y <- rnorm(20, mean=1)
# p-value obtained from a two-sample t-test
t.test(x,y, alternative="less")$p.value
# p-value obtained from an intra-experiment analysis
intraAnalysisClassic(x, y, func=t.test, alternative = "less")
# p-value obtained from a two-sample wilcoxon test
wilcox.test(x,y,alternative="less")$p.value
# p-value obtained from an intra-experiment analysis
intraAnalysisClassic(x, y, func= wilcox.test, alternative = "less")

intraAnalysisGene

Intra-experiment analysis of an expression dataset at the gene-level

Description

perform an intra-experiment analysis in conjunction with the moderated t-test (limma package) for the purpose of differential expression analysis of a gene expression dataset

Usage

intraAnalysisGene(data, group, splitSize = 5, metaMethod = addCLT)
intraAnalysisGene

Arguments

- **data**: a data frame where the rows are the gene IDs and the columns are the samples
- **group**: sample grouping. The elements of `group` are either 'c' (control) or 'd' (disease). `names(group)` should be identical to `colnames(data)`
- **splitSize**: the minimum number of disease samples in each split dataset. `splitSize` should be at least 3. By default, `splitSize=5`
- **metaMethod**: the method used to combine p-values. This should be one of `addCLT` (additive method [1]), `fishersMethod` (Fisher’s method [5]), `stoufferMethod` (Stouffer’s method [6]), `max` (maxP method [7]), or `min` (minP method [8])

Details

This function performs an intra-experiment analysis [1] for individual genes of the given dataset. The function first splits the dataset into smaller datasets, performs a moderated t-test (limma package) for the genes of the split datasets, and then combines the p-values for individual genes using `metaMethod`

Value

A data frame (rownames are gene IDs) that consists of the following information:

- `logFC`: log foldchange (diseases versus controls)
- `pLimma`: p-value obtained from limma without splitting
- `pLimma.fdr`: FDR-corrected p-values of `pLimma`
- `pIntra`: p-value obtained from intra-experiment analysis
- `pIntra.fdr`: FDR-corrected p-values of `pIntra`

Author(s)

Tin Nguyen and Sorin Draghici

References


See Also

`bilevelAnalysisGene`, `intraAnalysisClassic`, `link{bilevelAnalysisClassic}`

Examples

```r
data(GSE33223)
X <- intraAnalysisGene(data_GSE33223, group_GSE33223)
head(X)
```
loadKEGGPathways  

Load KEGG pathways and names

Description
Load KEGG pathways and names

Usage
loadKEGGPathways(organism = "hsa", updateCache = FALSE)

Arguments
organism  organism code. Default value is "hsa" (human)
updateCache  re-download KEGG pathways. Default value is FALSE

Value
A list of the following components

•  \textit{kpg} a list of \texttt{graphNEL} objects encoding the pathway information.
•  \textit{kpn} a named vector of pathway tiles. The names of the vector are the pathway KEGG IDs.

Author(s)
Tin Nguyen and Sorin Draghici

See Also
keggPathwayGraphs, keggPathwayNames

Examples
x <- loadKEGGPathways()

stoufferMethod  

Stouffer's method for meta-analysis

Description
Combine independent studies using the sum of p-values transformed into standard normal variables

Usage
stoufferMethod(x)

Arguments
\(x\)  is an array of independent p-values
Details

Considering a set of \( m \) independent significance tests, the resulted p-values are independent and uniformly distributed between 0 and 1 under the null hypothesis. Stouffer’s method is similar to Fisher’s method (`fisherMethod`), with the difference is that it uses the sum of p-values transformed into standard normal variables instead of the log product.

Value

combined p-value

Author(s)

Tin Nguyen and Sorin Draghici

References


See Also

`fisherMethod`, `addCLT`

Examples

```r
x <- rep(0, 10)
stoufferMethod(x)

x <- runif(10)
stoufferMethod(x)
```
Index

*Topic dataset
  GSE17054, 11
  GSE33223, 11
  GSE42140, 12
  GSE57194, 12

addCLT, 2, 10, 17

bilevelAnalysisClassic, 3, 14
bilevelAnalysisGene, 4, 4, 5, 14, 15
bilevelAnalysisGeneset, 5, 9
bilevelAnalysisPathway, 7, 8

data_GSE17054 (GSE17054), 11
data_GSE33223 (GSE33223), 11
data_GSE42140 (GSE42140), 12
data_GSE57194 (GSE57194), 12

fisherMethod, 3, 10, 17

graphNEL, 8, 16
group_GSE17054 (GSE17054), 11
group_GSE33223 (GSE33223), 11
group_GSE42140 (GSE42140), 12
group_GSE57194 (GSE57194), 12
GSA, 7
GSE17054, 11
GSE33223, 11
GSE42140, 12
GSE57194, 12

intraAnalysisClassic, 3–5, 13, 15
intraAnalysisGene, 4, 14, 14

keggPathwayGraphs, 16
keggPathwayNames, 16

loadKEGGPathways, 16

padog, 7
pe, 9
phyper, 7, 9

stoufferMethod, 3, 10, 16