Package 'Mergeomics'

November 12, 2026

Type Package

Title Integrative network analysis of omics data

Version 1.40.0

Date 2016-01-04

Author Ville-Petteri Makinen, Le Shu, Yuqi Zhao, Zeyneb Kurt, Bin Zhang, Xia Yang

Maintainer Zeyneb Kurt <zeynebkurt@gmail.com>

Description The Mergeomics pipeline serves as a flexible framework for integrating multidimensional omics-disease associations, functional genomics, canonical pathways and gene-gene interaction networks to generate mechanistic hypotheses. It includes two main parts,
1) Marker set enrichment analysis (MSEA);
2) Weighted Key Driver Analysis (wKDA).

biocViews Software

Suggests RUnit, BiocGenerics

License GPL (>= 2)

Depends R (>= 3.0.1)

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

git_branch RELEASE_3_23

git_last_commit 115b529

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

Mergeomics-package 3
job.kda 4
kda.analyze 5
kda.analyze.exec 7

kda.analyze.simulate . . . 9
kda.analyze.test . . . 11
kda.configure . . . 14
kda.finish . . . 16
kda.finish.estimate . . . 17
kda.finish.save . . . 18
kda.finish.summarize . . . 20
kda.finish.trim . . . 21
kda.prepare . . . 22
kda.prepare.overlap . . . 24
kda.prepare.screen . . . 26
kda.start . . . 28
kda.start.edges . . . 29
kda.start.identify . . . 31
kda.start.modules . . . 32
kda2cytoscape . . . 33
kda2cytoscape.colorize . . . 35
kda2cytoscape.colormap . . . 36
kda2cytoscape.drivers . . . 37
kda2cytoscape.edges . . . 39
kda2cytoscape.exec . . . 40
kda2cytoscape.identify . . . 42
kda2himmeli . . . 43
kda2himmeli.colorize . . . 45
kda2himmeli.colormap . . . 46
kda2himmeli.drivers . . . 47
kda2himmeli.edges . . . 48
kda2himmeli.exec . . . 50
kda2himmeli.identify . . . 52
MSEA.KDA.onestep . . . 53
ssea.analyze . . . 55
ssea.analyze.observe . . . 57
ssea.analyze.randgenes . . . 59
ssea.analyze.randloci . . . 62
ssea.analyze.simulate . . . 65
ssea.analyze.statistic . . . 67
ssea.control . . . 68
ssea.finish . . . 70
ssea.finish.details . . . 72
ssea.finish.fdr . . . 74
ssea.finish.genes . . . 76
ssea.meta . . . 78
ssea.prepare . . . 80
ssea.prepare.counts . . . 82
ssea.prepare.structure . . . 84
ssea.start . . . 86
ssea.start.configure . . . 88
ssea.start.identify . . . 91

ssea.start.relabel . . . 92
ssea2kda . . . 94
ssea2kda.analyze . . . 97
ssea2kda.import . . . 99
tool.aggregate . . . 101
tool.cluster . . . 102
tool.cluster.static . . . 103
tool.coalesce . . . 104
tool.coalesce.exec . . . 106
tool.coalesce.find . . . 107
tool.coalesce.merge . . . 108
tool.fdr . . . 109
tool.fdr.bh . . . 110
tool.fdr.empirical . . . 111
tool.graph . . . 112
tool.graph.degree . . . 113
tool.graph.list . . . 115
tool.metap . . . 117
tool.normalize . . . 118
tool.normalize.quality . . . 119
tool.overlap . . . 120
tool.read . . . 121
tool.save . . . 122
tool.subgraph . . . 123
tool.subgraph.find . . . 124
tool.subgraph.search . . . 125
tool.subgraph.stats . . . 126
tool.translate . . . 128
tool.unify . . . 129

Index **130**

`Mergeomics-package` *Integrative network analysis of omics data*

Description

The Mergeomics pipeline serves as a flexible framework for integrating multidimensional omics-disease associations, functional genomics, canonical pathways and gene-gene interaction networks to generate mechanistic hypotheses. It includes two main parts, 1) Marker set enrichment analysis (MSEA); 2) Weighted Key Driver Analysis (wKDA).

Details

Package:	Mergeomics
Type:	Package
Version:	1.1.10
Date:	2016-01-04
License:	GPL (>= 2)
Depends: R (>= 3.0.1)	
URL:	http://mergeomics.research.idre.ucla.edu/

Mergeomics amalgamates disease association information derived from multidimensional omics data (e.g., genome, epigenome, transcriptome, metablome) with functional genomics (e.g., eQTLs, ENCODE), canonical pathways (e.g., KEGG, Reactome), and molecular networks (e.g., gene regulatory networks, protein-protein interaction networks). Two main steps of the pipeline are: Marker set enrichment analysis (MSEA) and weighted key driver analysis (wKDA). MSEA takes the following data as input: i) disease association data (GWAS, EWAS, TWAS...), ii) functional genomics (eQTLs and/or ENCODE information), and iii) functionally related genes information extracted from knowledge-based biological pathways or data-driven network modules (e.g., coexpressed genes in a given tissue relevant to a disease of interest). These datasets are integrated via MSEA to return gene sets that are significantly enriched for markers showing low p value associations with a given disease. Then, the disease related gene sets are examined to detect the key drivers by using the wKDA step of the pipeline, which requires pre-defined directional networks such as tissue-specific Bayesian networks, protein-protein interaction networks, etc. wKDA maps the disease related gene sets to the pre-defined directional networks to identify key driver genes that are more likely regulators of the disease gene sets based on their central positions in the gene networks. The key drivers and their local network topology can be viewed and downloaded after the completion of the analysis via Visualization step. Our pipeline provides users to perform MSEA and wKDA together or separately using either their own input data or selecting preloaded sample datasets. The details of the functions and parameter settings are described in the Manual of the package.

Author(s)

Ville-Petteri Makinen, Le Shu, Yuqi Zhao, Zeyneb Kurt, Bin Zhang, Xia Yang Maintainer: <zeyneb@ucla.edu>

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

`job.kda` *Key Driver Analyzing results*

Description

Key Driver Analysis (KDA) applied data object. "modules.mousecoexpr.liver.human.txt" including the coexpression modules and "network.mouseliver.mouse.txt" including the network (graph) information files (under the extdata folder) were used for the KDA.

Format

The format is: list

Examples

```
data(job.kda)
```

`kda.analyze` *Weighted key driver analysis (wKDA) main function*

Description

Finds the statistics (enrichment score, p-value, FDR, etc.) of the key driver (hub) genes belonging to the specified modules based on the graph topology. The enrichment score of a hub node based on the shared nodes between this hub's neighbor nodes in the graph and the member nodes of the hub's module. The hub node enrichment P-values reflect the degree of observed enrichment of the hub, when compared to the null distribution of randomly expected enrichment of this hub within graph's nodes. Permutation test is used to obtain these statistics.

Usage

```
kda.analyze(job)
```

Arguments

`job`	The data list that will be subjected to KDA. It involves the modules, member genes belonging to each module, graph (network) topology, hubs of the graph, and sub-graph around each hub (hubnets of the graph).

Details

`kda.analyze` analyzes each module individually and determines the p-values and FDRs of hub nodes of each module via permutation test. It returns the hit hub (key driver) gene name and member list of each module.

Value

`job`	The KDA applied data list. It involves the modules, hub gene and member genes belonging to each module, and False Discovery Rate (FDR) adjusted p-values of hub nodes for each module.

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.analyze.exec, kda.analyze.simulate, kda.analyze.test

Examples

```
job.kda <- list()
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1
job.kda$nperm <- 20 # the default value is 2000, use 20 for unit tests

## kda.start() process takes long time while seeking hubs in the given net
## Here, we used a very small subset of the module list (1st 10 mods
## from the original module file):
moddata <- tool.read(job.kda$modfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
## save this to a temporary file and set its path as new job.kda$modfile:
tool.save(moddata, "subsetof.supersets.txt")
job.kda$modfile <- "subsetof.supersets.txt"

## Let's run KDA!

job.kda <- kda.configure(job.kda)
job.kda <- kda.start(job.kda)
job.kda <- kda.prepare(job.kda)
```

```
job.kda <- kda.analyze(job.kda)
job.kda <- kda.finish(job.kda)


## Remove the temporary files used for the test:
file.remove("subsetof.supersets.txt")
## remove the results folder
unlink("Results", recursive = TRUE)
```

---

kda.analyze.exec *Auxiliary function for weight key driver analysis (wKDA)*

Description

Obtains the enrichment scores (and p-values of these scores) of the hub nodes by the module member genes for a given module. The hub node enrichment P-values reflect the degree of enrichment of hub's neighbor nodes within the member genes of the module, to whom this hub belongs to, when compared to the null distribution of randomly expected enrichment of hub within graph's nodes.

Usage

```
kda.analyze.exec(memb, graph, nsim)
```

Arguments

memb	Member nodes of the given module.
graph	Entire graph (network) of the dataset.
nsim	Number of the simulations for the permutation test to obtain p-values of the enrichment scores belonging to the hub nodes for a given module.

Details

kda.analyze.exec obtains the p-values of the enrichment scores belonging to the hub nodes for a given module. Enrichment score of a hub node for a given module is obtained by the overlapped (shared) nodes between this hub's neighbor nodes and the member nodes of the given module. If a hub node does not have at least a particular number of neighbors, its enrichment score is assigned as 0.0.

Value

pvals	P-values of the enrichment scores belonging to the hub nodes for the given module.

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.analyze, kda.analyze.simulate, kda.analyze.test

Examples

```
## This auxiliary function is called by kda.analyze(),
## see this main function for more details
job.kda <- list()
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction<-1
job.kda$nperm <- 20 # the default value is 2000, use 20 for unit tests

## kda.start() process takes long time while seeking hubs in the given net
## Here, we used a very small subset of the module list (1st 10 mods
## from the original module file):
moddata <- tool.read(job.kda$modfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
## save this to a temporary file and set its path as new job.kda$modfile:
tool.save(moddata, "subsetof.supersets.txt")
job.kda$modfile <- "subsetof.supersets.txt"

## Let's prepare KDA object for KDA:
job.kda <- kda.configure(job.kda)
job.kda <- kda.start(job.kda)
job.kda <- kda.prepare(job.kda)
set.seed(job.kda$seed)
i = 1 ## index of the module, whose p-val is calculated:
memb <- job.kda$module2nodes[[i]]
```

```
graph <- job.kda$graph  ## we need to import a network
nsim <- job.kda$nperm   ## number of simulations
## calculate p-vals of KDs for the specified module:
# p <- kda.analyze.exec(memb, graph, nsim) ## see kda.analyze() for details

## Remove the temporary files used for the test:
file.remove("subsetof.supersets.txt")
## remove the results folder
unlink("Results", recursive = TRUE)
```

`kda.analyze.simulate` *Weighted key driver analysis (wKDA) simulation*

Description

Generates simulations for permutation test, which is performed to obtain the p-value for the enrichment score of a given hub for a specified module during the wKDA process.

Usage

```
kda.analyze.simulate(o, g, nmemb, nnodes, nsim)
```

Arguments

`o`	Observed enrichment score of a hub node assigned for a given module.
`g`	Sub-graph of a given hub and its neighbors (hubnet).
`nmemb`	Number of the members included in a given module.
`nnodes`	Number of the nodes in the whole graph (network) of the dataset.
`nsim`	Number of the iterations (simulations) performed for the permutation test.

Details

`kda.analyze.simulate` performs permutation tests to obtain p-values for the enrichment score of a given hub node for a given module. It takes the observed enrichment score of the given hub, hubnet (subgraph of the hub and its neighbors), number of the members of the given module, total number of the nodes in the entire graph of the dataset, and number of the simulations for the permutation test. In each iteration (simulation), it samples `nmemb` nodes randomly among the entire nodes of the graph. Then, it tests the overlapped nodes among the randomly chosen nodes and the given node's neigborhood. At the end, it obtains an enrichment score for each simulation and evaluates these permuted enrichment scores with respect to the observed enrichment score of the hub. Among `nsim` random simulations; maximally, enrichment scores of 10 iterations are allowed to be greater than the observed (actual) enrichment score of the hub. If this limitation is exceeded, simulation will be finalized at that point and the enrichment score list of the iterations will be returned.

Value

`x`	A list containing enrichment scores of the simulation's iterations

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.analyze, kda.analyze.exec, kda.analyze.test

Examples

```
job.kda <- list()
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction<-1
job.kda$nperm <- 20 # the default value is 2000, use 20 for unit tests

## kda.start() process takes long time while seeking hubs in the given net
## Here, we used a very small subset of the module list (1st 10 mods
## from the original module file):
moddata <- tool.read(job.kda$modfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
## save this to a temporary file and set its path as new job.kda$modfile:
tool.save(moddata, "subsetof.supersets.txt")
job.kda$modfile <- "subsetof.supersets.txt"

## Let's prepare KDA object for KDA:
job.kda <- kda.configure(job.kda)
job.kda <- kda.start(job.kda)
job.kda <- kda.prepare(job.kda)
set.seed(job.kda$seed)
```

```
i = 1 ## index of the module, whose p-val is calculated:
memb <- job.kda$module2nodes[[i]]
graph <- job.kda$graph  ## we need to import a network
nsim <- job.kda$nperm   ## number of simulations
## This auxiliary function is called by kda.analyze.exec(), which is called
## by kda.analyze() main function, see this main function for more details

hubs <- graph$hubs
hubnets <- graph$hubnets
nhubs <- length(hubs)
nnodes <- length(graph$nodes)
nmemb <- length(memb)

## Observed enrichment scores.
# obs <- rep(NA, nhubs)
# k <- 1 ## actual using: for(k in 1:nhubs){}, for unit test, use the 1st hub
# g <- hubnets[[hubs[k]]]
# obs[k] <- kda.analyze.test(g$RANK, g$STRENG, memb, nnodes)

## Estimate P-values.
# pvals <- rep(NA, nhubs)
# for(k in which(obs > 0)) {
# g <- hubnets[[hubs[k]]]
## First pass:
# x <- kda.analyze.simulate(obs[k], g, nmemb, nnodes, 200)
## Then, use x to estimate preliminary and final P-values.
## See kda.analyze() for more detail

## Remove the temporary files used for the test:
file.remove("subsetof.supersets.txt")
## remove the results folder
unlink("Results", recursive = TRUE)
# } ## finishing for loop
```

`kda.analyze.test` *Calculate enrichment score for wKDA*

Description

Obtains the enrichment score of a given hub (center node) belonging to a specified module. Enrichment score of a center node depends on the shared node number between the neighbor nodes of this center node (derived from the provided graph topology) and member nodes of this center node's module. The more a center node has neighbors in the graph among the member genes belonging to the module of this center node, the greater enrichment score it has.

Usage

```
kda.analyze.test(neigh, w, members, nnodes)
```

Arguments

neigh	Neighbor nodes of the given hub node (i.e. nodes in the hubnet)
w	Weigths of the given hub node based on its in-degree and out-degree edge density in the hubnet
members	Node indices -within the entire graph- of the member genes of given hub's module.
nnodes	Number of the nodes in the entire graph of the dataset.

Details

kda.analyze.test takes a hub node's neigbor list and weight list; additionally, it takes the member node list of relevant module. It searches the masses of the shared nodes between hubnet and the given module (gene set). The shared edge mass is normalized with respect to number of the expected match ratio between hubnet and the given node list. This normalized ratio is assigned as the observed enrichment score of the hubnet according to the given member node list.

Value

z	Calculated enrichment score

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.analyze, kda.analyze.exec, kda.analyze.simulate

Examples

```
job.kda <- list()
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<- "Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
```

```
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction<-1
job.kda$nperm <- 20 # the default value is 2000, use 20 for unit tests

## kda.start() process takes long time while seeking hubs in the given net
## Here, we used a very small subset of the module list (1st 10 mods
## from the original module file):
moddata <- tool.read(job.kda$modfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
## save this to a temporary file and set its path as new job.kda$modfile:
tool.save(moddata, "subsetof.supersets.txt")
job.kda$modfile <- "subsetof.supersets.txt"

## Let's prepare KDA object for KDA:
job.kda <- kda.configure(job.kda)
job.kda <- kda.start(job.kda)
job.kda <- kda.prepare(job.kda)
set.seed(job.kda$seed)
i = 1 ## index of the module, whose p-val is calculated:
memb <- job.kda$module2nodes[[i]]
graph <- job.kda$graph  ## we need to import a network
nsim <- job.kda$nperm   ## number of simulations
## This auxiliary function is called by kda.analyze.exec(), which is called
## by kda.analyze() main function, see this main function for more details

hubs <- graph$hubs
hubnets <- graph$hubnets
nhubs <- length(hubs)
nnodes <- length(graph$nodes)
nmemb <- length(memb)

## Observed enrichment scores for the hubs of the given module.
obs <- rep(NA, nhubs)
k <- 1 ## actual using: for(k in 1:nhubs){}, for test, use only the 1st hub
g <- hubnets[[hubs[k]]]
obs[k] <- kda.analyze.test(g$RANK, g$STRENG, memb, nnodes)

## Then, estimate preliminary and final P-values by kda.analyze.simulate()
## See kda.analyze() for more details

## Remove the temporary files used for the test:
file.remove("subsetof.supersets.txt")
## remove the results folder
unlink("Results", recursive = TRUE)
```

kda.configure *Set parameters for weighted key driver analysis (wKDA)*

Description

takes the configuration (plan) parameter for wKDA process as input and assigns default values if needed. The fields of this parameter are listed in the arguements section in detail.

Usage

```
kda.configure(plan)
```

Arguments

plan	a parameter including fields about the details of the wKDA process:

```
label:    unique identifier for the analysis
folder:   parent folder for results
netfile:  path to network file (TAIL HEAD WEIGHT)
modfile:  path to module file (MODULE GENE)
inffile:  path to module info file
nodfile:  path to node selection file
depthsearch: depth for subgraph search
direction: 0 for undirected, negative for downstream and
positive for upstream
maxoverlap: maximum allowed overlap between two key driver
neighborhoods
minsize:  minimum module size
mindegreeminimum: node degree to qualify as a hub
maxdegreemaximum: node degree to include
edgefactor: influence of node strengths: 0.0 no influence,
1.0 full influence
seed: seed for random number generator
```

Details

kda.configure prepares the environment for wKDA process, checks the fields of the input plan parameter (that includes paths of required input files and output folder, min module size, etc.), and assigns the default values to these fields if they are not specified.

Value

plan	configured and -if needed updated- plan parameter to be used in wKDA process.

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.analyze

Examples

```
## for KDA the essential parameters should be assigned by user is as follows:
plan <- list()
## assign job label:
plan$label<-"HDLC"
## specify parent folder for results:
plan$folder<-"Results"
## Get an input network (columns: TAIL HEAD WEIGHT)
plan$netfile <-"network.mouseliver.mouse.txt"
## Get the gene sets derived from ModuleMerge, containing two columns,
## MODULE and NODE, delimited by tab
plan$modfile<- "moddata.txt"
## If above parameters are not assigned by users, code will stop with error:
if(is.null(plan$folder)) stop("No parent folder.")
if(is.null(plan$label)) stop("No job label.")
if(is.null(plan$netfile)) stop("No network file.")
if(is.null(plan$modfile)) stop("No module file.")

## other parameters are optional, if they are not specified by user,
## kda.configure assigns their default values:
## graph search depth parameter:
if(is.null(plan$depth)) plan$depth <- 1
## edge directionality in the network: 0 means undirected
if(is.null(plan$direction)) plan$direction <- 0
## max overlap allowed between two modules
if(is.null(plan$maxoverlap)) plan$maxoverlap <- 0.33
## min size of the modules
if(is.null(plan$minsize)) plan$minsize <- 20
## min and max hub degree to be included:
if(is.null(plan$mindegree)) plan$mindegree <- "automatic"
if(is.null(plan$maxdegree)) plan$maxdegree <- "automatic"
## number of simulations for permutation test:
if(is.null(plan$nperm)) plan$nperm <- 2000
## seed for random number generator:
if(is.null(plan$seed)) plan$seed <- 1
## these are the main parameters needed to be assigned default values.
```

kda.finish *Organize and save results*

Description

After wKDA process is accomplished, kda.finish.estimate sums up the results and log them to the relevant files and folders. Besides, return them within the given job parameter.

Usage

```
kda.finish(job)
```

Arguments

job	the data list including label and folder fields to specify a unique identifier for the wKDA process and the output folder for the obtained results, respectively.

Details

kda.finish.estimate estimates additional measures if needed, saves results into relevant files, trims numbers to provide a simpler file for viewing, and stores a summary file of top hits after the wKDA prcess is accomplished. It also obtains the overlaps of the modules with hub neighborhoods, finds co-hubs information, determines the top key driver for each module and saves the updated and sorted p-values belonging to them.

Value

job	updated information including the overlapping hub neighborhoods, co-hubs information, top driver of each module, and their updated and sorted p-values.

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.finish.estimate, kda.finish.save, kda.finish.summarize, kda.finish.trim

Examples

```
## get the prepared and KDA applied dataset:(see kda.analyze for details)
data(job_kda_analyze)
## set the relevant parameters:
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1

## finish the KDA process
job.kda <- kda.finish(job.kda)

## remove the results folder
unlink("Results", recursive = TRUE)
```

---

kda.finish.estimate *Estimate measures for accomplished wKDA results*

Description

Estimates additional measures based on overlapping of module member nodes with hub neighbor nodes in the graph.

Usage

```
kda.finish.estimate(job)
```

Arguments

job	The data list that was subjected to wKDA. It involves the modules, member genes belonging to each module, graph information of the dataset, hubs and hubnets of the graph.

Details

kda.finish.save determines the overlaps of modules with hub neighborhoods, obtains graph measures based on the ratio of the observed overlap amounts to the expected overlap amount, and returns the values of this measure.

Value

res	Returns the overlapping ratio of the modules with hubnets.

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.finish, kda.finish.save, kda.finish.summarize, kda.finish.trim

Examples

```
## get the prepared and KDA applied dataset:(see kda.analyze for details)
data(job_kda_analyze)
## finish the KDA process by estimating additional measures for the modules
## such as module sizes, overlaps with hub neighborhoods, etc.
# job.kda <- kda.finish(job.kda)
# if (nrow(job.kda$results)==0){
# cat("No Key Driver Found!!!!")
# } else{
##  Estimate additional measures - see kda.analyze and kda.finish for details
#   res <- kda.finish.estimate(job.kda)
# }
```

---

kda.finish.save *Save full wKDA results*

Description

kda.finish.save sorts (according to KD p-values) and saves the wKDA results into specified files and folders.

Usage

```
kda.finish.save(res, job)
```

Arguments

res	the results obtained from kda.finish.estimate. They will be stored into specified folder.
job	information including the entire graph, nodes, modules, co-hubs, top key driver of each module, and their updated and sorted p-values. All the information included job will be stored into relevant files.

Value

res	the results obtained from kda.finish.estimate are merged with the module and graph nodes information gained from job data frame. At the end, merged information is both written to file and returned to the user.

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.finish, kda.finish.estimate, kda.finish.summarize, kda.finish.trim

Examples

```
## get the prepared and KDA applied dataset:(see kda.analyze for details)
data(job_kda_analyze)
## finish the KDA process by estimating additional measures for the modules
## such as module sizes, overlaps with hub neighborhoods, etc.
# job.kda <- kda.finish(job.kda)
# if (nrow(job.kda$results)==0){
# cat("No Key Driver Found!!!!")
# } else{
##  Estimate additional measures - see kda.analyze and kda.finish for details
#   res <- kda.finish.estimate(job.kda)
##  Save full results about modules such as co-hub, nodes, P-values info etc.
#   res <- kda.finish.save(res, job.kda)
# }
```

`kda.finish.summarize` *Summarize the wKDA results*

Description

Create a summary file of top key drivers. The file includes the key driver of each block of the dataset and their p-values.

Usage

```
kda.finish.summarize(res, job)
```

Arguments

`res`	the data frame including the p-values, false discovery rates, and fold scores of the nodes obtained from `kda.finish.trim`
`job`	the data frame including the path of output file which will briefly contain top key drivers of the blocks and ranked p-values of those top key drivers

Details

`kda.finish.summarize` determines the ranking scores of blocks, finds the top node for each block, selects and saves top key drivers, and stores P-values into file. top drovers of the blocks are also returned to the user.

Value

`res`	data frame including top node for each block

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

`kda.finish`, `kda.finish.estimate`, `kda.finish.save`, `kda.finish.trim`

Examples

```
## get the prepared and KDA applied dataset:(see kda.analyze for details)
data(job_kda_analyze)
## finish the KDA process by estimating additional measures for the modules
## such as module sizes, overlaps with hub neighborhoods, etc.
# job.kda <- kda.finish(job.kda)
# if (nrow(job.kda$results)==0){
# cat("No Key Driver Found!!!!")
# } else{
##  Estimate additional measures - see kda.analyze and kda.finish for details
#   res <- kda.finish.estimate(job.kda)
##  Save full results about modules such as co-hub, nodes, P-values info etc.
#   res <- kda.finish.save(res, job.kda)
##  Create a simpler file for viewing by trimming floating numbers
#   res <- kda.finish.trim(res, job.kda)
##  Create a summary file of top hit KDs.
#   res <- kda.finish.summarize(res, job.kda)
# }
## See kda.analyze() and kda.finish() for details
```

---

kda.finish.trim *Trim numbers before save*

Description

kda.finish.trim trims p-values, false discovery rates, and fold scores to make them nicer to look at before saving the file. It also returns trimmed results to the user.

Usage

```
kda.finish.trim(res, job)
```

Arguments

res	includes p-values, false discovery rates, and fold scores of the nodes
job	data frame including output folder path to store trimmed results

Value

res	Trimmed and formatted p-values, false discovery rates, and fold scores of the nodes

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.finish, kda.finish.estimate, kda.finish.save, kda.finish.summarize

Examples

```
## get the prepared and KDA applied dataset:(see kda.analyze for details)
data(job_kda_analyze)
## finish the KDA process by estimating additional measures for the modules
## such as module sizes, overlaps with hub neighborhoods, etc.
# job.kda <- kda.finish(job.kda)
# if (nrow(job.kda$results)==0){
# cat("No Key Driver Found!!!!")
# } else{
##  Estimate additional measures - see kda.analyze and kda.finish for details
#   res <- kda.finish.estimate(job.kda)
##  Save full results about modules such as co-hub, nodes, P-values info etc.
#   res <- kda.finish.save(res, job.kda)
##  Create a simpler file for viewing by trimming floating numbers
#   res <- kda.finish.trim(res, job.kda)
# }
## See kda.analyze() and kda.finish() for details
```

---

kda.prepare *Prepare graph topology for weighted key driver analysis*

Description

kda.prepare gets graph topology required by wKDA process, then provides the information including hub list, hubnets, and overlapping co-hubs.

Usage

```
kda.prepare(job)
```

Arguments

job	a parameter including restirictions while determining the graph topology information (such as hubs, hubnets, co-hubs, etc.), which is required by the wKDA process:

```
graph: graph of the dataset
depth: search depth for subgraph search
direction: use 0 for undirected, negative for downstream
and positive for upstream
maxoverlap: maximum allowed overlap between two key driver
neighborhoods
mindegree: minimum hub degree to include
edgefactor: influence of node strengths; 0.0 no influence,
1.0 full influence
```

Details

kda.prepare determines minimum hub degree if it is not specified by the user, finds hubs and their neighborhoods (hubnets), extracts overlapping co-hubs, returns this information to user, and prints it to the screen.

Value

job — Updated data frame including information about the graph topology in terms of hubs, hubnets, and overlapping co-hubs:

```
hubs: hub nodes list
hubnets: neighborhoods of hubs (hubnets)
cohubsets: overlapping hubs (co-hubs)
```

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.analyze, kda.prepare.overlap, kda.prepare.screen

Examples

```
job.kda <- list()
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<- "Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
```

```
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1
job.kda$nperm <- 20 # the default value is 2000, use 20 for unit tests

## kda.start() process takes long time while seeking hubs in the given net
## Here, we used a very small subset of the module list (1st 10 mods
## from the original module file):
moddata <- tool.read(job.kda$modfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
## save this to a temporary file and set its path as new job.kda$modfile:
tool.save(moddata, "subsetof.supersets.txt")
job.kda$modfile <- "subsetof.supersets.txt"

## Configure the parameters for KDA:
job.kda <- kda.configure(job.kda)
## Create the object properly
job.kda <- kda.start(job.kda)
## Find the hubs, co-hubs, and hub neighborhoods (hubnets), etc.:
job.kda <- kda.prepare(job.kda)
## After that, we need to call kda.analyze() and kda.finish()

## Remove the temporary files used for the test:
file.remove("subsetof.supersets.txt")
## remove the results folder
unlink("Results", recursive = TRUE)
```

`kda.prepare.overlap` *Extract overlapping co-hubs*

Description

`kda.prepare.overlap` finds overlapping co-hubs of the given graph.

Usage

```
kda.prepare.overlap(graph, direction, rmax)
```

Arguments

`graph`	entire graph, whose overlapping co-hubs will be found

direction	the direction of the interactions among graph components. 0 for undirected, negative for downstream, and positive for upstream
rmax	maximum allowed overlap between two key driver neighborhoods

Value

graph	Updated graph including overlapping co-hubs:
	cohubsetsco-hubs of the given graph

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.analyze, kda.prepare

Examples

```
job.kda <- list()
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<- "Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-"network.mouseliver.mouse.txt"
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- "mergedModules.txt"
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1

## Configure the parameters for KDA:
# job.kda <- kda.configure(job.kda)
## Create the object properly
# job.kda <- kda.start(job.kda)

## Find the hubs, co-hubs, and hub neighborhoods (hubnets) by kda.prepare()
## and its auxiliary functions kda.prepare.screen and kda.prepare.overlap
## First, determine the minimum and maximum hub degrees:
# nnodes <- length(job.kda$graph$nodes)
# if (job.kda$mindegree == "automatic") {
```

```
#   dmin <- as.numeric(quantile(job.kda$graph$stats$DEGREE,0.75))
#   job.kda$mindegree <- dmin
# }
# if (job.kda$maxdegree == "automatic") {
#   dmax <- as.numeric(quantile(job.kda$graph$stats$DEGREE,1))
#   job.kda$maxdegree <- dmax
# }
## Collect neighbors.
# job.kda$graph <- kda.prepare.screen(job.kda$graph, job.kda$depth,
# job.kda$direction, job.kda$edgefactor, job.kda$mindegree, job.kda$maxdegree)

## Then, extract overlapping co-hubs by kda.prepare.overlap():
## Collect overlapping co-hubs.
# job.kda$graph <- kda.prepare.overlap(job.kda$graph, job.kda$direction,
# job.kda$maxoverlap)
```

`kda.prepare.screen` *Prepare hubs and hubnets*

Description

`kda.prepare.screen` finds hubs and their neighborhoods (hubnets) from the given graph.

Usage

```
kda.prepare.screen(graph, depth, direction, efactor, dmin, dmax)
```

Arguments

`graph`	entire graph, whose hubs and hubnets will be obtained
`depth`	search depth for subgraph search
`direction`	the direction of the interactions among graph components. 0 for undirected, negative for downstream, and positive for upstream
`efactor`	influence of node strengths (weights): 0.0 no influence, 1.0 full influence
`dmin`	minimum hub degree to include
`dmax`	maximum hub degree to include

Value

`graph`	Updated graph including obtained hubs and hubnets:
	`hubs: hub nodes list` `hubnets: neighborhoods of hubs (hubnets)`

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.analyze, kda.prepare

Examples

```
job.kda <- list()
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<- "Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-"network.mouseliver.mouse.txt"
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- "mergedModules.txt"
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1

## Configure the parameters for KDA:
# job.kda <- kda.configure(job.kda)
## Create the object properly
# job.kda <- kda.start(job.kda)

## Find the hubs, co-hubs, and hub neighborhoods (hubnets) by kda.prepare()
## and its auxiliary functions kda.prepare.screen and kda.prepare.overlap
## First, determine the minimum and maximum hub degrees:
# nnodes <- length(job.kda$graph$nodes)
# if (job.kda$mindegree == "automatic") {
#   dmin <- as.numeric(quantile(job.kda$graph$stats$DEGREE,0.75))
#   job.kda$mindegree <- dmin
# }
# if (job.kda$maxdegree == "automatic") {
#   dmax <- as.numeric(quantile(job.kda$graph$stats$DEGREE,1))
#   job.kda$maxdegree <- dmax
# }
## Collect neighbors.
# job.kda$graph <- kda.prepare.screen(job.kda$graph, job.kda$depth,
# job.kda$direction, job.kda$edgefactor, job.kda$mindegree, job.kda$maxdegree)

## Then, extract overlapping co-hubs by kda.prepare.overlap()
```

kda.start *Import data for weighted key driver analysis*

Description

kda.start converts identities (such as module descriptions, module identifiers, and module nodes) to indices. It prepares graph topology and module information for wKDA process.

Usage

```
kda.start(job)
```

Arguments

job	a data frame including fields for edges and nodes information of the graph (TAIL, HEAD, WEIGHT). It also involves path of input files including module descriptions and module-gene lists.

Details

kda.start imports graph and relevant module descriptor input files, creates an indexed graph structure, and converts identities to indices from module descriptions and module-gene lists. Hence, it concludes with a graph structure and a module set involving member gene IDs for each module.

Value

job — Updated data frame including indexed graph topology, modules, and nodes information:

```
graph:      indexed topology
modules:    module identities
modinfo:    module descriptions (indexed)
moddata:    module data (indexed)
module2nodes:  lists of node indices for each module
modulesizes:   module sizes
```

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.analyze, kda.finish, kda.prepare, kda.start.edges, kda.start.identify, kda.start.modules

Examples

```
job.kda <- list()
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1
job.kda$nperm <- 20 # the default value is 2000, use 20 for unit tests

## kda.start() process takes long time while seeking hubs in the given net
## Here, we used a very small subset of the module list (1st 10 mods
## from the original module file):
moddata <- tool.read(job.kda$modfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
## save this to a temporary file and set its path as new job.kda$modfile:
tool.save(moddata, "subsetof.supersets.txt")
job.kda$modfile <- "subsetof.supersets.txt"

job.kda <- kda.configure(job.kda)
## Import data for weighted key driver analysis:
job.kda <- kda.start(job.kda)

## Remove the temporary files used for the test:
file.remove("subsetof.supersets.txt")
## remove the results folder
unlink("Results", recursive = TRUE)
```

`kda.start.edges` *Import nodes and edges of graph topology*

Description

`kda.start.edges` imports network file, gets edge data (in TAIL, HEAD, WEIGHT format), eliminates the nodes -whose degree is smaller than the maximum allowed node degree-, and returns the edges of remaining nodes.

Usage

```
kda.start.edges(job)
```

Arguments

job	a data frame including information such as network file name, maximum allowed node degree, edge direction (job$netfile, job$maxdegree, job$direction, and so on.)

Value

edgdata	filtered edge list, i.e. edges of the nodes, whose degree is smaller than the maximum allowed node degree

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.analyze, kda.finish, kda.prepare, kda.start

Examples

```
job.kda <- list()
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1
job.kda$nperm <- 20 # the default value is 2000, use 20 for unit tests
job.kda <- kda.configure(job.kda)
```

```
## Import topology of the graph for KDA
## This is already had been done in the kda.start() main function, due to
## the time constraint while running examples, we did not run it again.
# edgdata <- kda.start.edges(job.kda)

## remove the results folder
unlink("Results", recursive = TRUE)
```

`kda.start.identify` *Convert identities to indices for wKDA*

Description

`kda.start.identify` searches the members of `dat` among the members of `labels` with respect to the `varname` attribute, returns the matching rows of the `dat`.

Usage

```
kda.start.identify(dat, varname, labels)
```

Arguments

`dat`	data list of the identities that will be searched
`varname`	search will be performed with respect to which attribute (MODULE or NODE)
`labels`	the place, where data list (i.e. `dat`) will be searched

Value

`res`	matched rows of `dat` among the members of `labels` list according to the `varname` attribute

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

`kda.analyze`, `kda.finish`, `kda.prepare`, `kda.start`

Examples

```
## Converts identities (either module names or gene names) to the indices
aa<- data.frame(MODULE=c("Mod1", "Mod1", "Mod2", "Mod2", "Mod3"),
NODE=c("GeneA", "GeneC", "GeneB", "GeneC", "GeneA"))
aa
bb <- kda.start.identify(aa, "MODULE", c("Mod1"))
bb
cc <- kda.start.identify(aa, "MODULE", c("Mod1", "Mod3"))
cc
dd <- kda.start.identify(aa, "NODE", c("GeneA"))
dd
```

---

kda.start.modules *Import module descriptions*

Description

kda.start.modules searches the whole nodes of the modules within the nodes of edgdata edge-list, filters out the nodes that does not exist in the nodes of edgdata, and deletes the modules, which does not have enough nodes.

Usage

```
kda.start.modules(job, edgdata)
```

Arguments

job	a data frame including information such as module data file name, edge direction, minimum acceptable module size (job$modfile, job$direction, job$minsize, and so on.)
edgdata	edge list data obtained from kda.start.edges

Value

moddata	module descriptions and their member node lists

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.analyze, kda.finish, kda.prepare, kda.start

Examples

```
job.kda <- list()
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1
job.kda$nperm <- 20 # the default value is 2000, use 20 for unit tests
job.kda <- kda.configure(job.kda)

## Import topology of the graph for KDA, then find the module statistics
## This is already had been done in the kda.start() main function, due to
## the time constraint while running examples, we did not run it again.
# edgdata <- kda.start.edges(job.kda)
## Find module memberships of the graph nodes and obtain module statistics:
# moddata <- kda.start.modules(job.kda, edgdata)

## remove the results folder
unlink("Results", recursive = TRUE)
```

---

kda2cytoscape *Generate input files for Cytoscape*

Description

kda2cytoscape generates input files for Cytoscape to visualize the graph and hubnets after the wKDA process finished. The network visualization is a streamlined depiction of the module enrichment in hub neighborhoods.

Usage

```
kda2cytoscape(job, node.list = NULL, modules = NULL, ndrivers = 5,
depth = 1)
```

Arguments

job	wKDA result data list as returned by kda.finish
node.list	array of node/gene names to be visualized with their neighbor node. if this is not specified top ndrivers of each module and their neighborhoods will be illustrated.
modules	array of module names to be visualized
ndrivers	maximum number of drivers per module
depth	depth for neighborhood search in the graph

Details

kda2cytoscape first, selects top scoring key drivers for each module; then, assigns a colormap to modules, processes each module separately, finds key nodes' neighborhoods within a particular search depth, and saves the edge and node lists of the modules to the specified output folder. Besides, it returns this configuration data to the user. Created file list for Cytoscape are given below:

```
kda2cytoscape.top.kds.txt: top key drivers of the modules are
listed in this file. Number of the key drivers can be set by
user with ndrivers parameter.
kda2cytoscape.edges.txt: edge lists of the integrated graph
that includes the subnetworks of all modules.
kda2cytoscape.nodes.txt: node lists of the integrated graph
that includes the subnetworks of all modules.
module.color.mapping.txt: color mapping for the  modules,
i.e. one color is assigned to each module.
```

Value

job	updated data list including the node and edge information of the modules converted to Cytoscape format

Author(s)

Zeyneb Kurt

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.analyze, kda.finish

Examples

```
## get the prepared and KDA applied dataset:(see kda.analyze for details)
data(job_kda_analyze)
## set the relevant parameters:
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1

## finish the KDA process
job.kda <- kda.finish(job.kda)

## prepare the cytoscape-ready files:
job.kda <- kda2cytoscape(job.kda)

## remove the results folder
unlink("Results", recursive = TRUE)
```

---

kda2cytoscape.colorize

Trace module memberships of genes

Description

`kda2cytoscape.colorize` assigns color to each node of the given module. If a node belongs to more than one module, different colors will be assigned to that node, as each color representing one module (shared nodes are illustrated as pie charts in the graph).

Usage

```
kda2cytoscape.colorize(noddata, moddata, modpool, palette)
```

Arguments

noddata	node information of the entire graph
moddata	module data including node (member gene) list
modpool	unique module list including significant key drivers
palette	assigned unique color map for all modules

Value

res	data frame including the assigned color labels for the nodes of the given module. If a node is concurrently member of many modules, multiple colors will be assigned to that node (one color for each of these modules)

Author(s)

Zeyneb Kurt

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda2cytoscape

Examples

```
## Trace module memberships for each KD and its neighbors
## If a KD (and its neigbor nodes) is member of multiple modules, assign
## multiple colors to these multi-member nodes.
## We need to know data of all possible modules and all possible module ids
## to assign multiple colors to a shared node (between modules) when needed
if(exists("valdata"))
cat("Marker pvalues will be used to determine node sizes
in the network illustration")
# noddata <- kda2cytoscape.colorize(neighs, job.kda$moddata, modpool, palette)
```

---

kda2cytoscape.colormap

Assign one color to each unique module

Description

kda2cytoscape.colormap takes number of the modules and assigns a particular color to each module. Returns the color list (palette).

Usage

```
kda2cytoscape.colormap(ncolors)
```

Arguments

ncolors	number of the unique modules

Value

palette	color list: one color is assigned to each module

Author(s)

Zeyneb Kurt

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda2cytoscape

Examples

```
color.number = 5 ## let us assume we have 5 modules, assign 1 color to each:
palette <- kda2cytoscape.colormap(color.number)
```

---

kda2cytoscape.drivers *Select top key drivers for each module*

Description

kda2cytoscape.drivers finds maximally top ndriv key drivers for each module with respect to the significance level of the drivers.

Usage

```
kda2cytoscape.drivers(data, modules, ndriv)
```

Arguments

data	data frame including information of the modules (key driver list, p-values, node list, false discovery rates (fdr), and so on.)
modules	top scoring modules among KDA results
ndriv	maximum number of drivers that can be chosen for per module

Value

data	top key drivers (maximally ndriv drivers for each module) for top modules

Author(s)

Zeyneb Kurt

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda2cytoscape

Examples

```
## get the prepared and KDA applied dataset:(see kda.analyze for details)
data(job_kda_analyze)
## set the relevant parameters:
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1

## Finish the KDA process
job.kda <- kda.finish(job.kda)
## Select top key drivers from each module.
## First, take module names from kda results
modules <- unique(job.kda$results$MODULE)
## Take top 2 KDs:
drivers <- kda2cytoscape.drivers(job.kda$results, modules, ndriv=2)

## remove the results folder
unlink("Results", recursive = TRUE)
```

kda2cytoscape.edges *Find edges of a given node with a specified depth*

Description

kda2cytoscape.edges finds the sub-graph (node and edge lists) of a central node and its neighborhood at a particular search depth. The central node is a member of a module, which is defined at kda2cytoscape.exec.

Usage

```
kda2cytoscape.edges(graph, center, depth, direction)
```

Arguments

graph	entire graph
center	the node, whose interactions with neighbors will be searched within graph.
depth	search depth for graph neighborhood
direction	edge direction. 0 for undirected, negative for downstream and positive for upstream

Value

g	the sub-graph including TAIL, HEAD, WEIGHT information of the central node, which belongs to the specified module.

Author(s)

Zeyneb Kurt

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda2cytoscape

Examples

```
## get the prepared and KDA applied dataset:(see kda.analyze for details)
data(job_kda_analyze)
## set the relevant parameters:
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
```

```
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1

## Finish the KDA process
job.kda <- kda.finish(job.kda)
## Select a center node to seek its neighbors in the graph:
edges.of.center.node <- kda2cytoscape.edges(job.kda$graph, 1,
job.kda$depth, job.kda$direction)

## remove the results folder
unlink("Results", recursive = TRUE)
```

`kda2cytoscape.exec` *Evaluate each module separately for visualization*

Description

`kda2cytoscape.exec` deals with the modules individually; takes a particular amount of top key drivers of the given module in company with the top key driver lists and colormap of all modules; traces module memberships and produces colormap, it finds the edge and node lists for the top key drivers and their neighborhood for a given module.

Usage

```
kda2cytoscape.exec(job, drivers, modpool, palette, graph.depth = 1)
```

Arguments

`job`	data list including entire graph, nodes, modules information
`drivers`	top key drivers of the specified module
`modpool`	unique key driver list for all modules
`palette`	assigned unique color map for all modules
`graph.depth`	search depth for graph neighborhood

Value

res	uniquely identified node and edge lists of the members belonging to the given module

Author(s)

Zeyneb Kurt

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda2cytoscape

Examples

```
## get the prepared and KDA applied dataset:(see kda.analyze for details)
data(job_kda_analyze)
## set the relevant parameters:
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1

## Finish the KDA process
job.kda <- kda.finish(job.kda)
## Select top key drivers from each module.
## First, take module names from kda results
modules <- unique(job.kda$results$MODULE)
## Take top 2 KDs:
drivers <- kda2cytoscape.drivers(job.kda$results, modules, ndriv=2)
drivers <- as.data.frame(drivers)
colnames(drivers) <- c("MODULE" , "NODE")
```

```
mods <- unique(drivers$MODULE)
modnames <- job.kda$modules[mods]
modnames[which(mods == 0)] <- "NON.MODULE"
palette <- kda2cytoscape.colormap(length(mods))
palette[,which(mods == 0)] <- c(90,90,90)
drivers$MODNAMES <- modnames[match(drivers$MODULE, mods)]
drivers$NODNAMES <- job.kda$graph$nodes[drivers$NODE]
for(i in 1:nrow(drivers))
drivers$COLOR[i] <- paste(palette[1, match(drivers$MODULE[i], mods)],
palette[2, match(drivers$MODULE[i], mods)],
palette[3, match(drivers$MODULE[i], mods)], sep=" ")
## Process each module separately. Just perform for the 1st module:
i <- 1
rows <- which(drivers$MODULE == mods[i])
if(length(rows) > 0)
tmp <- kda2cytoscape.exec(job.kda, drivers[rows,], mods, palette,
job.kda$depth)

## remove the results folder
unlink("Results", recursive = TRUE)
```

`kda2cytoscape.identify` *Match identities with respect to given variable name*

Description

`kda2cytoscape.identify` searches the given data list `dat` within the `labels` according to the specified attribute (variable name). It returns the matched rows. Hence, it finds identifier numbers for the searched data list `dat`.

Usage

```
kda2cytoscape.identify(dat, varname, labels)
```

Arguments

`dat`	node ID list whose symbols or names will be collected from network node name (or symbol) list.
`varname`	specifies that `dat` will be searched among `labels` according to which variable (attribute). Here, gene symbols whose IDs are given, will be searched in the causal network node list according to the NODE attribute.
`labels`	the data list possibly including names or symbols corresponding to the given IDs in the `dat` data list.

Value

`res`	the matching rows of `labels` with the identifiers of given data list `dat`

Author(s)

Zeyneb Kurt

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda2cytoscape

Examples

```
## Converts identities (either module names or gene names) to the indices
aa<- data.frame(MODULE=c("Mod1", "Mod1", "Mod2", "Mod2", "Mod3"),
NODE=c("GeneA", "GeneC", "GeneB", "GeneC", "GeneA"))
aa
bb <- kda2cytoscape.identify(aa, "MODULE", c("Mod1"))
bb
cc <- kda2cytoscape.identify(aa, "MODULE", c("Mod1", "Mod3"))
cc
dd <- kda2cytoscape.identify(aa, "NODE", c("GeneA"))
dd
```

---

kda2himmeli *Generate input files for Himmeli*

Description

kda2himmeli generates input files for Himmeli to visualize the graph and hubnets after the wKDA process finished. The network visualization is a streamlined depiction of the module enrichment in hub neighborhoods.

Usage

```
kda2himmeli(job, modules = NULL, ndrivers = 5)
```

Arguments

job	KDA result data list as returned by kda.finish
modules	array of module names to be visualized
ndrivers	maximum number of drivers per module

Details

kda2himmeli first, selects top scoring key drivers for each module; then, assigns a colormap to modules, processes each module separately, finds key nodes' neighborhoods, and saves the edge and node lists of the modules to the specified output folder. Besides, it returns this configuration data to the user.

Value

job	updated data list including the node and edge information of the modules converted to Himmeli format

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda.analyze, kda.finish

Examples

```
## get the prepared and KDA applied dataset:(see kda.analyze for details)
data(job_kda_analyze)
## set the relevant parameters:
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
job.kda$nodfile <- system.file("extdata","msea2kda.nodes.txt",
package="Mergeomics")

## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1
```

```
## finish the KDA process
job.kda <- kda.finish(job.kda)

## prepare the cytoscape-ready files:
job.kda <- kda2himmeli(job.kda)

## remove the results folder
unlink("Results", recursive = TRUE)
```

`kda2himmeli.colorize` *Trace module memberships of genes*

Description

`kda2himmeli.colorize` assigns color to each node of the given module. If a node belongs to more than one module, different colors will be assigned to that node, as each color representing one module (shared nodes are illustrated as pie charts in the graph).

Usage

```
kda2himmeli.colorize(noddata, moddata, modpool, palette)
```

Arguments

`noddata`	node information of the entire graph
`moddata`	module data including node (member gene) list
`modpool`	unique module list including significant key drivers
`palette`	assigned unique color map for all modules

Value

`res`	data frame including the assigned color labels for the nodes of the given module. If a node is concurrently member of many modules, many colors will be assigned to that node (one color for each of these modules)

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda2himmeli

Examples

```
## Trace module memberships for each KD
## If a KD is member of multiple modules, assign multiple colors to it
## Also consider the locus pval of the top locus of each KD (by valdata)
## We need to know data of all possible modules and all possible module ids
## to assign multiple colors(sectors) to a KD when needed
if(exists("valdata"))
cat("Marker pvalues will be used to determine node sizes
in the network illustration")
# noddata <- kda2himmeli.colorize(valdata, job.kda$moddata, modpool, palette)
```

---

kda2himmeli.colormap *Assign one color to each unique module*

Description

kda2himmeli.colormap takes number of the modules and assigns a particular color to each module. Returns the color list (palette).

Usage

```
kda2himmeli.colormap(ncolors)
```

Arguments

ncolors	number of the unique modules

Value

palette	color list: one color is assigned to each module

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda2himmeli

Examples

```
color.number = 5 ## let us assume we have 5 modules, assign 1 color to each:
palette <- kda2himmeli.colormap(color.number)
```

---

kda2himmeli.drivers *Select top key drivers for each module*

Description

kda2himmeli.drivers finds maximally top ndriv key drivers for each module with respect to the significance level of the drivers.

Usage

```
kda2himmeli.drivers(data, modules, ndriv)
```

Arguments

data	data frame including information of the modules (key driver list, p-values, node list, false discovery rates (fdr), and so on.)
modules	top scoring modules among KDA results
ndriv	maximum number of drivers that can be chosen for per module

Value

data	top key drivers (maximally ndriv drivers for each module) for top modules (if module significance levels are given)

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda2himmeli

Examples

```
## get the prepared and KDA applied dataset:(see kda.analyze for details)
data(job_kda_analyze)
## set the relevant parameters:
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
job.kda$nodfile <- system.file("extdata","msea2kda.nodes.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1

## Finish the KDA process
job.kda <- kda.finish(job.kda)
## Select top key drivers from each module.
## First, take module names from kda results
modules <- unique(job.kda$results$MODULE)
## Take top 2 KDs:
drivers <- kda2himmeli.drivers(job.kda$results, modules, ndriv=2)

## remove the results folder
unlink("Results", recursive = TRUE)
```

`kda2himmeli.edges` *Find edges of a given node with a specified depth*

Description

`kda2himmeli.edges` finds the sub-graph (node and edge lists) of a central node and its neighborhood at a particular search depth. The central node is a member of a module, which is defined at `kda2himmeli.exec`.

Usage

```
kda2himmeli.edges(graph, center, depth, direction)
```

Arguments

graph	entire graph
center	the node, whose interactions with neighbors will be searched within graph.
depth	search depth for graph neighborhood
direction	edge direction. 0 for undirected, negative for downstream and positive for upstream

Value

g	the sub-graph including TAIL, HEAD, WEIGHT information of the central node, which belongs to the specified module.

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda2himmeli

Examples

```
## get the prepared and KDA applied dataset:(see kda.analyze for details)
data(job_kda_analyze)
## set the relevant parameters:
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
job.kda$nodfile <- system.file("extdata","msea2kda.nodes.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
```

```
job.kda$direction <- 1

## Finish the KDA process
job.kda <- kda.finish(job.kda)
## Select a center node to seek its neighbors in the graph:
edges.of.center.node <- kda2himmeli.edges(job.kda$graph, 1,
job.kda$depth, job.kda$direction)

## remove the results folder
unlink("Results", recursive = TRUE)
```

`kda2himmeli.exec` *Evaluate each module separately for visualization*

Description

`kda2himmeli.exec` deals with the modules individually; takes a particular amount of top key drivers of the given module in company with the top key driver lists and colormap of all modules; traces module memberships and produces colormap, it finds the edge and node lists for the top key drivers and their neighborhood for a given module.

Usage

```
kda2himmeli.exec(job, valdata, drivers, modpool, palette)
```

Arguments

`job`	data list including entire graph, nodes, modules information
`valdata`	GWAS pvalues of top loci of the nodes - if this information is available, sizes of the nodes in the figure will be correlated with the p-value of the top loci of the nodes -
`drivers`	top key drivers of the specified module
`modpool`	unique key driver list for all modules
`palette`	assigned unique color map for all modules

Value

`res`	uniquely identified node and edge lists of the members belonging to the given module

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda2himmeli

Examples

```
## get the prepared and KDA applied dataset:(see kda.analyze for details)
data(job_kda_analyze)
## set the relevant parameters:
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## Gene sets derived from ModuleMerge, containing two columns, MODULE,
## NODE, delimited by tab
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
job.kda$nodfile <- system.file("extdata","msea2kda.nodes.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1

## Finish the KDA process
job.kda <- kda.finish(job.kda)

## Get valdata including marker pvals
valdata <- tool.read(job.kda$nodfile)
z <- as.double(valdata$VALUE)
z <- (z/quantile(z, 0.95) + rank(z)/length(z))
valdata$SIZE <- pmin(4.0, z)
## Select subset of genes.
valdata <- kda2himmeli.identify(valdata, "NODE", job.kda$graph$nodes)

## Select top key drivers from each module.
## First, take module names from kda results
modules <- unique(job.kda$results$MODULE)
## Take top 2 KDs:
drivers <- kda2himmeli.drivers(job.kda$results, modules, ndriv=2)
drivers <- as.data.frame(drivers)
colnames(drivers) <- c("MODULE" , "NODE")

mods <- unique(drivers$MODULE)
modnames <- job.kda$modules[mods]
modnames[which(mods == 0)] <- "NON.MODULE"
palette <- kda2himmeli.colormap(length(mods))
```

```
palette[,which(mods == 0)] <- c(90,90,90)
drivers$MODNAMES <- modnames[match(drivers$MODULE, mods)]
drivers$NODNAMES <- job.kda$graph$nodes[drivers$NODE]
for(i in 1:nrow(drivers))
drivers$COLOR[i] <- paste(palette[1, match(drivers$MODULE[i], mods)],
palette[2, match(drivers$MODULE[i], mods)],
palette[3, match(drivers$MODULE[i], mods)], collapse=" ")
## Process each module separately. Just perform for the 1st module:
i <- 1
rows <- which(drivers$MODULE == mods[i])
if(length(rows) > 0)
tmp <- kda2himmeli.exec(job.kda, valdata, drivers[rows,], mods, palette)

## remove the results folder
unlink("Results", recursive = TRUE)
```

`kda2himmeli.identify` *Match identities with respect to given variable name*

Description

`kda2himmeli.identify` searches the given data list `dat` within the `labels` according to the specified attribute (variable name). It returns the matched rows. Hence, it finds identifier numbers for the searched data list `dat`.

Usage

```
kda2himmeli.identify(dat, varname, labels)
```

Arguments

`dat`	node ID list whose symbols or names will be collected from network node name (or symbol) list.
`varname`	specifies that `dat` will be searched among `labels` according to which variable (attribute). Here, gene symbols whose IDs are given, will be searched in the causal network node list according to the NODE attribute.
`labels`	the data list possibly including names or symbols corresponding to the given IDs in the `dat` data list.

Value

`res`	the matching labels or names of `labels` with the IDs of `dat` list

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

kda2himmeli

Examples

```
## Converts identities (either module names or gene names) to the indices
aa<- data.frame(MODULE=c("Mod1", "Mod1", "Mod2", "Mod2", "Mod3"),
NODE=c("GeneA", "GeneC", "GeneB", "GeneC", "GeneA"))
aa
bb <- kda2himmeli.identify(aa, "MODULE", c("Mod1"))
bb
cc <- kda2himmeli.identify(aa, "MODULE", c("Mod1", "Mod3"))
cc
dd <- kda2himmeli.identify(aa, "NODE", c("GeneA"))
dd
```

MSEA.KDA.onestep *Run MSEA and/or KDA in one step*

Description

MSEA.KDA.onestep performs Marker Set Enrichment Analysis (MSEA) and/or Key Driver Anlaysis (KDA) processes in one step.

Usage

```
MSEA.KDA.onestep(plan, apply.MSEA=TRUE, apply.KDA=FALSE,
maxoverlap.genesets=0.33, symbol.transfer.needed=FALSE,
sym.from=c("HUMAN", "MOUSE"), sym.to=c("HUMAN", "MOUSE"))
```

Arguments

plan	a data list including file and parameter settings for MSEA and/or KDA processes: label: unique identifier for the analysis folder: output folder for results modfile: path to module file (cols: MODULE GENE) genfile: path to gene file (cols: GENE LOCUS) (MSEA-specific) marfile: path to marker file (cols: MARKER VALUE) (MSEA-specific) inffile: path to module info file (cols: MODULE DESCR) seed: seed for random number generator

```
permtype: gene for gene-level, locus for marker-level
nperm: max number of random permutations
mingenes: min number of genes per module (after merging)
maxgenes: max number of genes per module
quantiles: cutoffs for test statistic
maxoverlap: max overlap allowed between genes
netfile:  path to network file (TAIL HEAD WEIGHT) (KDA-specific)
```

`apply.MSEA` determines whether MSEA will be performed to the given set. Default value is TRUE.

`apply.KDA` determines whether KDA will be performed to the given set. Default value is FALSE.

`maxoverlap.genesets`
maximum overlapping ratio for the genesets. This is applicable if KDA is performed following the MSEA process in one-step running. Default value is 0.33.

`symbol.transfer.needed`
determines whether gene symbols in the gene sets are needed to be transformed between different species. Default value is FALSE.

`sym.from` defines the species, whose gene symbols will be converted to the gene symbols of `sym.to` species. It can be either HUMAN or MOUSE. It is applicable if `symbol.transfer.needed` is TRUE.

`sym.to` defines the species, whose gene symbols will be converted from the gene symbols of `sym.from` species. It can be either HUMAN or MOUSE. It is applicable if `symbol.transfer.needed` is TRUE.

Details

`MSEA.KDA.onestep` performs MSEA and/or KDA operations in one built-in function. Users can run both MSEA and KDA sequentially, or they can run either MSEA or KDA in one step with the same function. If MSEA and KDA will be applied sequentially, significantly enriched gene sets (having FDR < 0.25), coming from MSEA results, will be merged if their overlapping ratios are larger than a given threshold, i.e. `maxoverlap.genesets`, to proceed the next step with relatively indepent gene sets. Then, KDA is applied to this relatively independent gene sets.

Value

`plan` the updated data frame after performing MSEA and/or KDA. If MSEA is performed, results will include standard MSEA results (see `ssea.analyze` for details); if KDA is applied, results will include standard KDA results (see `kda.analyze` for details).

Author(s)

Zeyneb Kurt

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

ssea.analyze, kda.analyze

Examples

```
plan <- list()
plan$label <- "hdlc"
plan$folder <- "Results"
plan$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
plan$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
plan$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
plan$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
plan$nperm <- 100 ## default value is 20000
plan <- MSEA.KDA.onestep(plan, apply.MSEA=TRUE)
```

---

ssea.analyze *Marker set enrichment analysis (MSEA)*

Description

ssea.analyze finds the enrichment of the pathways or co-expression modules by a marker set (e.g. associated risk variants -loci- of a relevant disease). Association study by mapping markers (e.g. SNPs) to genes (e.g. via expression QTLs). Enrichment P-values obtained by the MSEA denote the degree of enrichment of significantly disease-associated (high ranking) markers (e.g. eSNPs) within these pathways when compared to the null distribution of expected uniform distribution of all ranks of the markers. MSEA is performed with either gene-level or marker-level permutations based on Gaussian distribution.

Usage

```
ssea.analyze(job, trim_start, trim_end)
```

Arguments

job	the data list including fields: seed for random number generator (job$seed) (to obtain the same results from permutation when the same input set is given), random permutation level (job$permtype) (either gene- or marker-level), maximum number of permutations (job$nperm) for the permutation test, and the database that uses indexed identities for modules, genes, and markers (e.g. loci) (job$database).
trim_start	percentile taken from the beginning for trimming away a defined proportion of genes with significant trait association to avoid signal inflation of null background in gene permutation. Default value is 0.002.

`trim_end` percentile taken from the ending point for trimming away a defined proportion of genes with significant trait association to avoid signal inflation of null background in gene permutation. Default value is 0.998.

Details

`ssea.analyze` associates the gene sets (pathways or co-expression modules) with relevant disease (e.g. Coronary Artery Disease) association data by mapping markers (e.g. SNPs) to genes (e.g. via expression QTLs). It performs the MSEA by using observed and estimated enrichment scores. First, the observed enrichment scores of the pathways by markers (e.g. loci) are calculated. Then, a Gaussian distribution based simulation is performed, by using the statistics of the observed scores (mean, std.dev., etc.), to obtain the estimated enrichment scores, enrichment frequencies, and other statistics e.g. p-values for the pathways. `ssea.analyze` trims away a defined proportion of genes with significant trait association to avoid signal inflation of null background in gene permutation by using `trim_start` and `trim_end`.

Value

`job` the updated data frame including results: indexed module identity, enrichment P-values, raw frequencies (raw frequency of a gene set defines the number of the estimated enrichment scores that are larger than this gene set's enrichment score under the null distribution based on Gaussian function)

.

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

`ssea.control`, `ssea.finish`, `ssea.prepare`, `ssea.start`, `ssea2kda`

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
```

```
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
job.msea <- ssea.start(job.msea)
job.msea <- ssea.prepare(job.msea)
job.msea <- ssea.control(job.msea)
job.msea <- ssea.analyze(job.msea)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea.analyze.observe` *Collect enrichment score statistics for MSEA*

Description

`ssea.analyze.observe` obtains the observed enrichment scores of the pathways or modules by a given marker set (e.g. GWAS loci data of a disease) depending on the observation frequencies of this markers in the pathways.

Usage

```
ssea.analyze.observe(db)
```

Arguments

db	database including the indexed identities for modules, genes and marker:
	`modulesizes: gene counts for modules.`
	`modulelengths: distinct marker counts for modules.`
	`moduledensities: ratio between distinct and non-distinct markers.`
	`genesizes: marker count for each gene.`
	`module2genes: gene lists for each module.`
	`gene2loci: marker lists for each gene.`
	`locus2row: row indices in the marker data frame for each marker.`
	`observed: matrix of observed counts of values that exceed each quantile point for each marker.`
	`expected: 1.0 - quantile points.`

Value

scores	enrichment scores

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

ssea.analyze

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
```

```
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"

## run ssea.start() and prepare for this small set: (due to the huge runtime)
job.msea <- ssea.start(job.msea)
job.msea <- ssea.prepare(job.msea)
job.msea <- ssea.control(job.msea)

## Observed enrichment scores.
db <- job.msea$database
scores <- ssea.analyze.observe(db)
nmods <- length(scores)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea.analyze.randgenes` *Estimate enrichment from randomized genes*

Description

`ssea.analyze.randgenes` simulates enrichment scores by randomizing the genes from all modules (from database - db)

Usage

```
ssea.analyze.randgenes(db, targets, gene_sel)
```

Arguments

db	database including the indexed identities for modules, genes and markers: modulesizes: gene counts for modules. modulelengths: distinct marker counts for modules. moduledensities: ratio between distinct and non-distinct markers. genesizes: marker count for each gene. module2genes: gene lists for each module. gene2loci: marker lists for each gene . locus2row: row indices in the marker data frame for each marker. observed: matrix of observed counts of values that exceed each quantile point for each marker. expected: 1.0 - quantile points.
targets	all modules
gene_sel	selected genes to be trimmed away to avoid signal inflation of null background in gene permutation.

Value

scores	randomly simulated enrichment scores

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

ssea.analyze

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
```

```
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
job.msea <- ssea.start(job.msea)
job.msea <- ssea.prepare(job.msea)
job.msea <- ssea.control(job.msea)

## Observed enrichment scores.
db <- job.msea$database
gene2loci <- db$gene2loci
locus2row <- db$locus2row
observed <- db$observed
#Calcuate individual gene enrichment score
trim_scores <- rep(NA, length(gene2loci))
for(k in 1:length(trim_scores)) {
  genes <- k
  # Collect markers.
  loci <- integer()
  for(i in genes)
    loci <- c(loci, gene2loci[[i]])

  # Determine data rows.
  loci <- unique(loci)
  rows <- locus2row[loci]
  nloci <- length(rows)

  # Calculate total counts.
  e <- (nloci/length(locus2row))*colSums(observed)
  o <- observed[rows,]
  if(nloci > 1) o <- colSums(o)
```

```
  # Estimate enrichment.
  trim_scores[k] <- ssea.analyze.statistic(o, e)
}
trim_start=0.002 # default
trim_end=1-trim_start
cutoff=as.numeric(quantile(trim_scores,probs=c(trim_start,trim_end)))
gene_sel=which(trim_scores>cutoff[1]&trim_scores<cutoff[2])

scores <- ssea.analyze.observe(db)
nmods <- length(scores)

## Simulated scores.
nperm <- job.msea$nperm
observ <- scores
## Include only non-empty modules for simulation.
nmods <- length(db$modulesizes)
targets <- which(db$modulesizes > 0)
hits <- rep(NA, nmods)
hits[targets] <- 0

## Prepare data structures to hold null samples.
keys <- rep(0, nperm)
scores <- rep(NA, nperm)
scoresets <- list()
for(i in 1:nmods) scoresets[[i]] <- double()
## Simulate random scores.
## within a for loop: check capacity, find new statistics, update snull
## distribution (simulated null distr.) by permuting genes
snull <- ssea.analyze.randgenes(db, targets, gene_sel)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea.analyze.randloci` *Estimate enrichment from randomized marker*

Description

`ssea.analyze.randloci` simulates enrichment scores by randomizing the marker that mapped to genes from all modules (from database, db)

Usage

```
ssea.analyze.randloci(db, targets)
```

Arguments

db	database including the indexed identities for modules, genes and markers: modulesizes: gene counts for modules. modulelengths: distinct marker counts for modules. moduledensities: ratio between distinct and non-distinct markers. genesizes: marker count for each gene. module2genes: gene lists for each module. gene2loci: marker lists for each gene . locus2row: row indices in the marker data frame for each marker. observed: matrix of observed counts of values that exceed each quantile point for each marker. expected: 1.0 - quantile points.
targets	all modules

Value

scores	randomly simulated enrichment scores

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

ssea.analyze

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000
```

```
## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
job.msea <- ssea.start(job.msea)
job.msea <- ssea.prepare(job.msea)
job.msea <- ssea.control(job.msea)

## Observed enrichment scores.
db <- job.msea$database
scores <- ssea.analyze.observe(db)
nmods <- length(scores)

## Simulated scores.
nperm <- job.msea$nperm
observ <- scores
## Include only non-empty modules for simulation.
nmods <- length(db$modulesizes)
targets <- which(db$modulesizes > 0)
hits <- rep(NA, nmods)
hits[targets] <- 0

## Prepare data structures to hold null samples.
keys <- rep(0, nperm)
scores <- rep(NA, nperm)
scoresets <- list()
for(i in 1:nmods) scoresets[[i]] <- double()
## Simulate random scores.
## within a for loop: check capacity, find new statistics, update snull
## distribution (simulated null distr.) by permuting loci
snull <- ssea.analyze.randloci(db, targets)

## Remove the temporary files used for the test:
```

```
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea.analyze.simulate` *Simulate scores for MSEA*

Description

`ssea.analyze.simulate` simulates enrichment scores by randomly permuting database with respect to the specified permutation type (either gene-level or marker-level).

Usage

```
ssea.analyze.simulate(db, observ, nperm, permtype, trim_start, trim_end)
```

Arguments

db	database including the indexed identities for modules, genes and markers (e.g. loci):
	`modulesizes: gene counts for modules.`
	`modulelengths: distinct marker counts for modules.`
	`moduledensities: ratio between distinct and non-distinct markers.`
	`genesizes: marker count for each gene.`
	`module2genes: gene lists for each module.`
	`gene2loci: marker lists for each gene.`
	`locus2row: row indices in the marker data frame for each marker.`
	`observed: matrix of observed counts of values that exceed each quantile point for each marker.`
	`expected: 1.0 - quantile points.`

observ	observed enrichment scores
nperm	maximum nubmer of permutations (for simulation)
permtype	permutation type (either gene or locus)
trim_start	percentile taken from the beginning for trimming away a defined proportion of genes with significant trait association to avoid signal inflation of null background in gene permutation. Default value is 0.002.
trim_end	percentile taken from the ending point for trimming away a defined proportion of genes with significant trait association to avoid signal inflation of null background in gene permutation. Default value is 0.998.

Value

scoresets	simulated score lists for the statistically significant modules

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

ssea.analyze

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
```

```
job.msea <- ssea.start(job.msea)
job.msea <- ssea.prepare(job.msea)
job.msea <- ssea.control(job.msea)

## Observed enrichment scores.
db <- job.msea$database
scores <- ssea.analyze.observe(db)
nmods <- length(scores)

## Simulated scores.
nperm <- job.msea$nperm
trim_start=0.002 # default
trim_end=1-trim_start
nullsets <- ssea.analyze.simulate(db, scores, nperm, job.msea$permtype,
trim_start, trim_end)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea.analyze.statistic`

MSEA statistics for enrichment score

Description

`ssea.analyze.statistic` estimates the enrichment score based on observed and expected ones.

Usage

```
ssea.analyze.statistic(o, e)
```

Arguments

o	observed enrichment score
e	expected enrichment score

Value

score	estimated enrichment score based on observed and expected scores

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

ssea.analyze

Examples

```
## O and E the observed and expected counts of positive findings
## (enrichment scores) at a given cutoff:
set.seed(1)
o <- rnorm(1)
e <- rnorm(1)
## find the final enrichment score from the observed and estimated scores:
z <- ssea.analyze.statistic(o, e)
```

---

ssea.control *Add internal positive control modules for MSEA*

Description

ssea.control adds positive control modules that includes the top-scored genes based on the marker scores of these genes. The database structure, including identities of the variables, is updated properly.

Usage

```
ssea.control(job)
```

Arguments

job	data list including module and gene identities as characters; also including database that has indexed identities for MSEA:
	`modules: module identities as characters.`
	`genes: gene identities as characters.`
	`moddata: preprocessed module data (indexed identities).`
	`database: database including indexed identities for modules, genes, and markers.`

Value

job	data list including augmented internal control modules:
	`modules: augmented module names`
	`moddata: augmented module data`
	`database: augmented database`

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

Examples

```
## Check the slots for control module;
## if it cannot find any control module, function throws an error,
## if can find control slots, updates the database identities (modules,
## genes, markers) properly:
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
```

```
job.msea <- ssea.start(job.msea)
job.msea <- ssea.prepare(job.msea)
job.msea <- ssea.control(job.msea)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea.finish` *Organize and save MSEA results*

Description

`ssea.finish` organizes and stores the MSEA results into relevant output files.

Usage

```
ssea.finish(job)
```

Arguments

`job`	data list including the results of MSEA process. `job` will be saved after getting organized:
	`label: unique identifier for the analysis.`
	`folder: output folder for results.`
	`resultsdata: frame including indexed module identities (MODULE) and enrichment P-values (P).`
	`database: database including indexed identities for modules, genes, and markers.`

Details

`ssea.finish` obtains module statistics (member genes, size, length, density, enrichment scores, false discovery rates), finds the top marker within genes, updates the gene scores and gene sizes (i.e. number of markers for each gene), and saves the organized results regarding the modules and genes into the relevant files.

Value

`job`	data list including the organized results of MSEA process:
	`results: updated information of modules:`
	`number of distinct member genes (NGENES),`
	`number of distinct member markers (NLOCI),`
	`ratio of distinct to non-distinct markers (DENSITY),`
	`false discovery rates (FDR).`
	`generesults: updated gene-specific information including:`

```
indexed gene identity (GENE),
gene size (NLOCI),
unadjusted enrichment score (SCORE),
marker with maximum value (LOCUS),
marker value (VALUE).
```

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

ssea.analyze, ssea.control, ssea.prepare, ssea.start, ssea2kda

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
```

```
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
job.msea <- ssea.start(job.msea)
job.msea <- ssea.prepare(job.msea)
job.msea <- ssea.control(job.msea)
job.msea <- ssea.analyze(job.msea)
job.msea <- ssea.finish(job.msea)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea.finish.details` *Organize and save module, gene, top locus, Ps of MSEA results*

Description

`ssea.finish.details` finds significant modules and their gene lists, and top marker (with GWAS -log10 transformed p-vals) of these genes, merge results of markers, genes and module statistics, sort results according to first, module enrichment score, then marker P-value, and saves these sorted results into the relevant files.

Usage

```
ssea.finish.details(job)
```

Arguments

`job`	data list including the results of MSEA process:

```
label: unique identifier for the analysis.
folder: output folder for results.
modinfo: descriptions of the modules.
resultsdata: frame including indexed module identities
(MODULE) and enrichment P-values (P).
database: database including indexed identities for
modules, genes, and markers.
```

Value

None.

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

ssea.finish

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
```

```
job.msea <- ssea.start(job.msea)
job.msea <- ssea.prepare(job.msea)
job.msea <- ssea.control(job.msea)
job.msea <- ssea.analyze(job.msea)
job.msea <- ssea.finish(job.msea)

## Estimate mod FDR values, sort according to significance, save full results:
job.msea <- ssea.finish.fdr(job.msea)
## Collect top markers(e.g.loci) within genes, save genes with top marker Pval
job.msea <- ssea.finish.genes(job.msea)
## Find signficant modules, collect gene members of top modules,
## Merge gene results (with top marker info),
## Sort and save details according to enrichment and marker value:
job.msea <- ssea.finish.details(job.msea)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea.finish.fdr` *Organize and save FDR results of the MSEA*

Description

`ssea.finish.fdr` estimates the FDR values of the enrichment P-values belonging to the modules. It also gets the other module information such as size (gene number), length (marker number), density, etc., sorts the modules according to P-values, saves this information into relevant files.

Usage

```
ssea.finish.fdr(job)
```

Arguments

`job`	data list including module-realted results of MSEA process:

	`folder: output folder for results.`
	`modules: module names.`
	`results: data frame including indexed module identities (MODULE) and enrichment P-values (P).`
	`database: database including indexed identities for modules, genes, and markers.`

Value

`job`	data list including the organized module-related results of MSEA process:

```
results: updated information of modules:
number of distinct member genes (NGENES),
number of distinct member markers (NLOCI),
ratio of distinct to non-distinct markers (DENSITY),
false discovery rates (FDR).
```

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

ssea.finish

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
```

```
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
job.msea <- ssea.start(job.msea)
job.msea <- ssea.prepare(job.msea)
job.msea <- ssea.control(job.msea)
job.msea <- ssea.analyze(job.msea)
job.msea <- ssea.finish(job.msea)

## Estimate mod FDR values, sort according to significance, save full results:
job.msea <- ssea.finish.fdr(job.msea)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea.finish.genes` *Organize and save gene-realted MSEA results*

Description

`ssea.finish.genes` organizes and stores the gene-related MSEA results into relevant output file. It finds the top markers within genes, update gene scores and gene sizes, and save the results.

Usage

```
ssea.finish.genes(job)
```

Arguments

`job`	data list including the information about the MSEA process:
	`folder: output folder for results.`
	`database: database including indexed identities for modules, genes, and markers.`

Value

`job`	data list including the organized gene-related results of MSEA process:
	`generesults: updated gene-specific information;`
	`indexed gene identity (GENE),`
	`gene size (NLOCI),`
	`unadjusted enrichment score (SCORE),`
	`marker with maximum value (LOCUS),`
	`marker value (VALUE).`

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

ssea.finish

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
```

```
job.msea <- ssea.start(job.msea)
job.msea <- ssea.prepare(job.msea)
job.msea <- ssea.control(job.msea)
job.msea <- ssea.analyze(job.msea)
job.msea <- ssea.finish(job.msea)

## Estimate mod FDR values, sort according to significance, save full results:
job.msea <- ssea.finish.fdr(job.msea)
## Collect top markers(e.g.loci) within genes, save genes with top marker Pval
job.msea <- ssea.finish.genes(job.msea)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea.meta` *Merge multiple MSEA results into meta MSEA*

Description

`ssea.meta` merges MSEA results of modules, genes, and markers, constructs hierarchical representation of genes and modules, calculates meta P-values of the modules (based on z-scores), and save all statistics results.

Usage

```
ssea.meta(jobs, label, folder)
```

Arguments

`jobs`	data list including information and statistics about genes, markers, and modules
`label`	label (unique identifier) for meta job
`folder`	parent folder for meta job

Value

`meta`	data list including meta-analyzing results for the modules, which enables analyzing the multiple MSEA results for the modules.

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

Examples

```
## Create an object for multiple MSEAs:
job.multiple.msea <- list()
set.seed(1)
for(i in 1:3){
## make 3 trials, each time pick 10 random modules among the first 20 modules
mod.indices <- sample(20, 10)
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 30 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[mod.indices]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
job.msea <- ssea.start(job.msea)
job.msea <- ssea.prepare(job.msea)
job.msea <- ssea.control(job.msea)
job.msea <- ssea.analyze(job.msea)
job.msea <- ssea.finish(job.msea)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
```

```
file.remove("subsetof.marfile.txt")
job.multiple.msea[[i]] <- job.msea
}

meta.results <- ssea.meta(job.multiple.msea, job.multiple.msea[[1]]$label,
job.multiple.msea[[1]]$folder)
```

---

ssea.prepare *Prepare an indexed database for MSEA*

Description

ssea.prepare prepares a database that includes hierarchical for modules, i.e. it collects gene list and unique marker list of the modules for MSEA process

Usage

```
ssea.prepare(job)
```

Arguments

job	a data list with the following components:

```
modules: module identities as characters.
genesgene: identities as characters.
loci: marker identities as characters.
moddata: preprocessed module data (indexed identities).
gendata: preprocessed mapping data (indexed identities).
locdata: preprocessed marker data (indexed identities).
mingenes: minimum module size allowed.
maxgenes: maximum module size allowed.
maxoverlap: maximum module overlap allowed (1.0 to skip).
quantiles: quantile points for test statistic.
```

Details

ssea.prepare removes extreme-sized modules, constructs a hierarchical representation of genes and modules, obtains hit counts for markers, and returns the finalized module, genes, markers, database information.

Value

job	an updated data list with the following components:

```
modules: finalized module names.
moddata: finalized module data.
gendata: finalized mapping data.
locdata: finalized marker data.
quantiles: verified quantile points.
```

```
database$modulesizes: gene counts for modules.
database$modulelengths: distinct markers counts for
modules.
database$moduledensities: ratio between distinct and
non-distinct markers.
database$genesizeslocus: count for each gene.
database$module2genes: gene lists for each module.
database$gene2locilocus: lists for each gene.
database$locus2row: indices in the marker data frame
for each marker.
database$observed: matrix of observed counts of values
that exceed each quantile point for each marker.
database$expected: 1.0 - quantile points.
```

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

ssea.analyze, ssea.control, ssea.finish, ssea.start, ssea2kda

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
```

```
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
job.msea <- ssea.start(job.msea)
job.msea <- ssea.prepare(job.msea)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea.prepare.counts` *Calculate hit counts up to a given quantile*

Description

Counts unique loci in a module, maps the marker data of a module to the all available markers by creating a bit matrix for values above the given quantiles. Created bit matrix contains either TRUE (above quantiles) or FALSE (below or equals to quantiles) values as a resuls of these comparisons. It returns the results (marker mapping and bit matrix)

Usage

```
ssea.prepare.counts(locdata, nloci, quantiles)
```

Arguments

`locdata`	marker data
`nloci`	number of elements in markers list
`quantiles`	quantile points for test statistic

Value

`res`	a data list with the following components:
	`locus2row: mapped marker information`
	`observed: bit matrix that involves TRUEs and FALSEs`

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

ssea.prepare

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
```

```
job.msea <- ssea.start(job.msea)

## Remove extremely big or small modules:
st <- tool.aggregate(job.msea$moddata$MODULE)
mask <- which((st$lengths >= job.msea$mingenes) &
(st$lengths <= job.msea$maxgenes))
pos <- match(job.msea$moddata$MODULE, st$labels[mask])
job.msea$moddata <- job.msea$moddata[which(pos > 0),]

## Construct hierarchical representation for modules, genes, and markers:
ngens <- length(job.msea$genes)
nmods <- length(job.msea$modules)
db <- ssea.prepare.structure(job.msea$moddata, job.msea$gendata,
nmods, ngens)
## Determine test cutoffs:
if(is.null(job.msea$quantiles)) {
lengths <- db$modulelengths
mu <- median(lengths[which(lengths > 0)])
job.msea$quantiles <- seq(0.5, (1.0 - 1.0/mu), length.out=10)
}
## Calculate hit counts:
nloci <- length(job.msea$loci)
hits <- ssea.prepare.counts(job.msea$locdata, nloci, job.msea$quantiles)
db <- c(db, hits)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea.prepare.structure` *Construct hierarchical representation of components*

Description

`ssea.prepare.structure` represents modules, genes, and markers in a hierarchical structure.

Usage

```
ssea.prepare.structure(moddata, gendata, nmods, ngens)
```

Arguments

`moddata`	module data (indexed identities)
`gendata`	mapping data (indexed identities)
`nmods`	number of modules
`ngens`	number of all genes

Details

ssea.prepare.structure finds member genes of modules and marker lists of genes; counts distinct markers within each module and obtains module's density from this count; at the end, it returns hierarchically structured results.

Value

res — a data list with the following components:

```
modulesizes: module size
modulelengths: module length
moduledensities: module densities
genesizes: gene sizes of module
module2genesgene: list of module
gene2loci: markers lists of genes
```

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

ssea.prepare

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
```

```
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
job.msea <- ssea.start(job.msea)

## Remove extremely big or small modules:
st <- tool.aggregate(job.msea$moddata$MODULE)
mask <- which((st$lengths >= job.msea$mingenes) &
(st$lengths <= job.msea$maxgenes))
pos <- match(job.msea$moddata$MODULE, st$labels[mask])
job.msea$moddata <- job.msea$moddata[which(pos > 0),]

## Construct hierarchical representation for modules, genes, and markers:
ngens <- length(job.msea$genes)
nmods <- length(job.msea$modules)
db <- ssea.prepare.structure(job.msea$moddata, job.msea$gendata,
nmods, ngens)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea.start` *Create a job for MSEA*

Description

Creates identities (for modules, member genes, and loci) to start MSEA process.

Usage

```
ssea.start(plan)
```

Arguments

plan	a data list with the following components:

```
label: unique identifier for the analysis
folder: output folder for results
modfile: path to module file (cols: MODULE GENE)
marfile: path to marker file (cols: MARKER VALUE)
genfile: path to gene file (cols: GENE LOCUS)
inffile: path to module info file (cols: MODULE DESCR)
seed: seed for random number generator
permtype: gene for gene-level, locus for marker-level
nperm: max number of random permutations
mingenes: min number of genes per module (after merging)
maxgenes: max number of genes per module
quantiles: cutoffs for test statistic
maxoverlap: max overlap allowed between genes
```

Details

`ssea.start` imports modules, genes-locus mapping, and locus values; removes the genes with no locus values from the list, find identities for modules, genes, loci components, and excludes missing data and factorize identities for these components.

Value

job	a data list with the following components:

```
modules: module identities as characters.
genes: gene identities as characters.
loci: marker identities as characters.
moddata: preprocessed module data (indexed identities)
modinfo: description of the modules.
gendata: preprocessed mapping data between genes and
markers (indexed identities).
locdata: preprocessed marker data (indexed identities)
geneclusters: genes with shared markers.
```

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

`ssea.analyze`, `ssea.control`, `ssea.finish`, `ssea.prepare`, `ssea2kda`

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() for this small set:(due to the huge runtime we did not use
## full sets of modules, genes, and markers)
job.msea <- ssea.start(job.msea)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea.start.configure` *Check parameters before MSEA*

Description

ssea.start.configure checks the input parameter before MSEA process starts and assigns default values for non-exist fields of the input data object.

Usage

```
ssea.start.configure(plan)
```

Arguments

plan	a data list with the following components: label: unique identifier for the analysis folder: output folder for results modfile: path to module file (cols: MODULE GENE) marfile: path to marker file (cols: MARKER VALUE) genfile: path to gene file (cols: GENE MARKER) inffile: path to module info file (cols: MODULE DESCR) seed: seed for random number generator permtype: gene for gene-level, marker for marker-level nperm: max number of random permutations mingenes: min number of genes per module (after merging) maxgenes: max number of genes per module quantiles: cutoffs for test statistic maxoverlap: max overlap allowed between genes

Value

plan	a data list including checked and assigned values (to non-existing fields) of the input parameter: label: unique identifier for the analysis folder: output folder for results modfile: path to module file (cols: MODULE GENE) marfile: path to marker file (cols: MARKER VALUE) genfile: path to gene file (cols: GENE MARKER) inffile: path to module info file (cols: MODULE DESCR) seed: seed for random number generator permtype: gene for gene-level, marker for marker-level nperm: max number of random permutations mingenes: min number of genes per module (after merging) maxgenes: max number of genes per module quantiles: cutoffs for test statistic maxoverlap: max overlap allowed between genes

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

ssea.start

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() for this small set:(due to the huge runtime we did not use
## full sets of modules, genes, and markers)
job.msea <- ssea.start.configure(job.msea)

## Remove the temporary files used for the test:
```

```
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea.start.identify` *Convert identities to indices for MSEA*

Description

`ssea.start.identify` finds matching identities for the given variable name. It searches the members of `dat` among the members of `labels` with respect to the `varname` attribute, returns the matching rows of the `dat`.

Usage

```
ssea.start.identify(dat, varname, labels)
```

Arguments

`dat`	data list (source) of the identities that will be searched. e.g. the information after merging of overlapped genes (containing shared markers)
`varname`	search and match will be performed with respect to which attribute (MODULE or NODE or MARKER)
`labels`	the place, where the identities of `dat` will be searched and matched.

Value

`res`	matched rows of `dat` among the members of `labels` list according to the `varname` attribute

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

`ssea.start`

Examples

```
## Converts identities (either module names or gene names) to the indices
aa<- data.frame(MODULE=c("Mod1", "Mod1", "Mod2", "Mod2", "Mod3"),
NODE=c("GeneA", "GeneC", "GeneB", "GeneC", "GeneA"))
aa
bb <- ssea.start.identify(aa, "MODULE", c("Mod1"))
bb
cc <- ssea.start.identify(aa, "MODULE", c("Mod1", "Mod3"))
cc
dd <- ssea.start.identify(aa, "NODE", c("GeneA"))
dd
```

---

ssea.start.relabel *Update gene symbols after merging overlapped markers*

Description

`ssea.start.relabel` updates gene symbols within the modules after merging overlapping genes that contain shared markers

Usage

```
ssea.start.relabel(dat, grp)
```

Arguments

`dat`	module data corresponding gene sets
`grp`	gene data that is needed to be relabeled after the merging process of the overlapping markers

Value

`dat`	relabeled module data of `grp`

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

`ssea.start`

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() for this small set:(due to the huge runtime we did not use
## full sets of modules, genes, and markers)
job.msea <- ssea.start.configure(job.msea)

## Import moddata:
moddata <- tool.read(job.msea$modfile, c("MODULE", "GENE"))
moddata <- unique(na.omit(moddata))

## Import marker (e.g. locus) values:
locdata <- tool.read(job.msea$locfile, c("LOCUS", "VALUE"))
locdata$VALUE <- as.double(locdata$VALUE)
rows <- which(0*(locdata$VALUE) == 0)
locdata <- unique(na.omit(locdata[rows,]))
locdata_ex <- locdata
names(locdata_ex) <- c("MARKER","VALUE")
```

```
## Import mapping data between genes and markers:
gendata <- tool.read(job.msea$genfile, c("GENE", "LOCUS"))
gendata <- unique(na.omit(gendata))
gendata_ex <- gendata
names(gendata_ex) <- c("GENE","MARKER")

## Remove genes with no marker values:
pos <- match(gendata$LOCUS, locdata$LOCUS)
gendata <- gendata[which(pos > 0),]

## Merge overlapping genes:
gendata <- tool.coalesce(items=gendata$LOCUS, groups=gendata$GENE,
rcutoff=job.msea$maxoverlap)
job.msea$geneclusters <- gendata[,c("CLUSTER","GROUPS")]
job.msea$geneclusters <- unique(job.msea$geneclusters)

## Update gene symbols after merging the overlapping ones:
moddata <- ssea.start.relabel(moddata, gendata)
gendata <- unique(gendata[,c("GROUPS", "ITEM")])
names(gendata) <- c("GENE", "LOCUS")

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

---

ssea2kda *Generate inputs for wKDA*

Description

ssea2kda forwards MSEA results to weighted key driver analysis (wKDA) from the first MSEA results, merges the overlapped modules according to a given overlapping ratio to obtain a relatively independent module set, apply a second MSEA on the merged modules (supersets), updates and saves the second MSEA results properly for wKDA process.

Usage

```
ssea2kda(job, symbols = NULL, rmax = NULL, min.module.count=NULL)
```

Arguments

job	data list including the organized results of MSEA process. It has following components:
	`results: updated information of modules including:`
	`number of distinct member genes (NGENES),`
	`number of distinct member markers (NLOCI),`
	`ratio of distinct to non-distinct markers (DENSITY),`
	`false discovery rates (FDR).`

```
                generesults: updated gene-specific information including:
                indexed gene identity (GENE),
                gene size (NLOCI),
                unadjusted enrichment score (SCORE),
                marker with max value (LOCUS),
                marker value (VALUE).
```

`symbols` dataframe for translating gene symbols

`rmax` maximum allowed overlap ratio between gene sets

`min.module.count`
minimum number of the pathways to be taken from the MSEA results to merge. Default value is NULL. If it is not specified, all the pathways having MSEA-FDR value less than 0.25 will be considered for merging if they are overlapping with the given ratio `rmax`.

Details

`ssea2kda` gets genes and top markers from input files, selects significant modules with respect to ordered p-values, gets identities of modules and genes, merges and trims the overlapping modules (either having FDR less than 0.25 or top `min.module.count` modules when ranked up to the P-values), obtains enrichment scores for merged modules, translates the gene symbols (between species) if needed, and finally saves the module, gene, node, and marker information into relevant output files.

Value

`plan` an updated data list with the following components:

```
label: unique identifier for the analysis.
parent: parent folder for results.
modfile: path of module file (columns: MODULE NODE).
inffile: path of module information file
(columns: MODULE DESCR).
nodfile: path of node selection file (columns: NODE).
```

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

`ssea.analyze`, `kda.analyze`

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
job.msea <- ssea.start(job.msea)
job.msea <- ssea.prepare(job.msea)
job.msea <- ssea.control(job.msea)
job.msea <- ssea.analyze(job.msea)
job.msea <- ssea.finish(job.msea)

###############  Create intermediary datasets for KDA ##################
syms <- tool.read(system.file("extdata", "symbols.txt",
package="Mergeomics"))
syms <- syms[,c("HUMAN", "MOUSE")]
names(syms) <- c("FROM", "TO")
job.ssea2kda <- ssea2kda(job.msea, symbols=syms)

## Remove the temporary files used for the test:
```

```
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea2kda.analyze` *Apply second MSEA after merging the modules*

Description

`ssea2kda.analyze` performs a second MSEA for the updated modules after merging the highly overlapped modules (according to a specified overlapping ratio)

Usage

```
ssea2kda.analyze(job, moddata)
```

Arguments

`job`	the data list including the information of modules, genes, and markers, and also involving the database that uses indexed identities for modules, genes, and markers (`job$database`).
`moddata`	merged modules including MODULE, GENE, and OVERLAP information

Details

`ssea2kda.analyze` constructs new gene lists for merged modules and updates module database including module sizes, lengths, densities (based on marker sizes), and gene list. Then, it runs a second MSEA and returns the enrichment scores of the updated module database.

Value

`res`	data list including updated information (after merge) such as, enrichment scores of merged modules

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

`ssea2kda`

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
unique(gendata$MARKER)))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
job.msea <- ssea.start(job.msea)
job.msea <- ssea.prepare(job.msea)
job.msea <- ssea.control(job.msea)
job.msea <- ssea.analyze(job.msea)
job.msea <- ssea.finish(job.msea)

###############  Create intermediary datasets for KDA ##################
syms <- tool.read(system.file("extdata", "symbols.txt",
package="Mergeomics"))
syms <- syms[,c("HUMAN", "MOUSE")]
names(syms) <- c("FROM", "TO")
## Collect genes and top markers from original files.
noddata <- ssea2kda.import(job.msea$genfile, job.msea$locfile)
```

```
## Select candidate modules (significant ones according to FDRs)
res <- job.msea$results
res <- res[order(res$P),]
rows <- which(res$FDR < 0.25)
res <- res[rows,]

## Collect member genes.
moddata <- job.msea$moddata
pos <- match(moddata$MODULE, res$MODULE)
moddata <- moddata[which(pos > 0),]

## Restore original identities.
modinfo <- job.msea$modinfo
modinfo$MODULE <- job.msea$modules[modinfo$MODULE]
moddata$MODULE <- job.msea$modules[moddata$MODULE]
moddata$GENE <- job.msea$genes[moddata$GENE]

## Merge and trim overlapping modules.
moddata$OVERLAP <- moddata$MODULE
rmax <- 0.33
moddata <- tool.coalesce(items=moddata$GENE, groups=moddata$MODULE,
rcutoff=rmax)
moddata$MODULE <- moddata$CLUSTER
moddata$GENE <- moddata$ITEM
moddata$OVERLAP <- moddata$GROUPS
moddata <- moddata[,c("MODULE", "GENE", "OVERLAP")]
moddata <- unique(moddata)

## Calculate enrichment scores for merged modules.
tmp <- unique(moddata[,c("MODULE","OVERLAP")])
res <- ssea2kda.analyze(job.msea, moddata)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`ssea2kda.import` *Import genes and top markers from original files*

Description

`ssea2kda.import` gets marker values from marker information file and mapping data (between genes and markers) from gene file, merges the imported information, and returns the merged data for top significant markers.

Usage

```
ssea2kda.import(genfile, locfile)
```

Arguments

genfile	gene information file
locfile	marker information file

Value

data	merged gene and corresponding marker data for top significant markers

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

See Also

ssea2kda

Examples

```
job.msea <- list()
job.msea$label <- "hdlc"
job.msea$folder <- "Results"
job.msea$genfile <- system.file("extdata",
"genes.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$marfile <- system.file("extdata",
"marker.hdlc_040kb_ld70.human_eliminated.txt", package="Mergeomics")
job.msea$modfile <- system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics")
job.msea$inffile <- system.file("extdata",
"coexpr.info.txt", package="Mergeomics")
job.msea$nperm <- 100 ## default value is 20000

## ssea.start() process takes long time while merging the genes sharing high
## amounts of markers (e.g. loci). it is performed with full module list in
## the vignettes. Here, we used a very subset of the module list (1st 10 mods
## from the original module file) and we collected the corresponding genes
## and markers belonging to these modules:
moddata <- tool.read(job.msea$modfile)
gendata <- tool.read(job.msea$genfile)
mardata <- tool.read(job.msea$marfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
gendata <- gendata[which(!is.na(match(gendata$GENE,
unique(moddata$GENE)))),]
mardata <- mardata[which(!is.na(match(mardata$MARKER,
```

```
unique(gendata$MARKER))),]

## save this to a temporary file and set its path as new job.msea$modfile:
tool.save(moddata, "subsetof.coexpr.modules.txt")
tool.save(gendata, "subsetof.genfile.txt")
tool.save(mardata, "subsetof.marfile.txt")
job.msea$modfile <- "subsetof.coexpr.modules.txt"
job.msea$genfile <- "subsetof.genfile.txt"
job.msea$marfile <- "subsetof.marfile.txt"
## run ssea.start() and prepare for this small set: (due to the huge runtime)
job.msea <- ssea.start(job.msea)
job.msea <- ssea.prepare(job.msea)
job.msea <- ssea.control(job.msea)
job.msea <- ssea.analyze(job.msea)
job.msea <- ssea.finish(job.msea)

###############  Create intermediary datasets for KDA ##################
syms <- tool.read(system.file("extdata", "symbols.txt",
package="Mergeomics"))
syms <- syms[,c("HUMAN", "MOUSE")]
names(syms) <- c("FROM", "TO")
## Collect genes and top markers from original files.
noddata <- ssea2kda.import(job.msea$genfile, job.msea$locfile)

## Remove the temporary files used for the test:
file.remove("subsetof.coexpr.modules.txt")
file.remove("subsetof.genfile.txt")
file.remove("subsetof.marfile.txt")
```

`tool.aggregate` *Aggregate the entries*

Description

`tool.aggregate` aggregates the entries with respect to the given feature. It first finds raw indices (either genes or markers), then sorts them, and finds the blocks (segments) of identical entries.

Usage

```
tool.aggregate(entries, limit = 1)
```

Arguments

`entries`	an array that will be sorted and aggregated within blocks
`limit`	minimum block size to be included

Value

res — a data list with the following components:

```
labels: shared values within blocks
lengths: numbers of entries in blocks
blocks: integer arrays of entry positions within blocks
ranks: entry positions included in blocks
```

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

Examples

```
aa <- data.frame(MODULE=c("Mod1", "Mod1", "Mod2", "Mod2", "Mod3"),
GENE=c("GeneA", "GeneC", "GeneB", "GeneC", "GeneA"))
## aggregate according to the module names:
bb <- tool.aggregate(aa$MODULE)
bb
## aggregate according to the gene names:
cc <- tool.aggregate(aa$GENE)
cc
```

---

tool.cluster *Hierarchical clustering of nodes*

Description

tool.cluster performs agglomerative hierarchical clustering for nodes (genes)

Usage

```
tool.cluster(edges, cutoff = NULL)
```

Arguments

edges	edge (weight) list among two group, whose overlapping information (overlapping ratio based on shared entries of two groups, number of members in both group) had been assesed previously
cutoff	cutting level of dendrogram for hierarchical clustering

Details

tool.cluster takes overlapping information between two groups, produces distance matrix based on 1-strength(overlap) ratio between two groups, and apply agglomerative hierarchical clustering based on the distance matrix.

Value

res — data list including clustering results:

```
CLUSTER: cluster label
NODE: item (node) name
```

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

Examples

```
## read the coexpr module file as an example:
moddata <- tool.read(system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics"))

## let us cluster the first 10 modules in the module file:
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
## Find clusters.
rmax = 0.33
edges <- tool.overlap(items=moddata$GENE, groups=moddata$MODULE)
clustdat <- tool.cluster(edges, cutoff=rmax)
nclust <- length(unique(clustdat$CLUSTER))
nnodes <- length(unique(clustdat$NODE))
```

---

tool.cluster.static *Static hierarchical clustering*

Description

tool.cluster.static takes dendrogram (clustering tree) and its cutting height; then, obtains cluster labels for the nodes of the tree.

Usage

```
tool.cluster.static(dendro, hlim)
```

Arguments

dendro	dendrogram (tree)
hlim	cutting height of the dendrogram

Value

clusters	cluster labels of the components after static clustering

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

Examples

```
set.seed(1)
## assume that we have a dataset including several samples
## with distinct features
dataset <- matrix(rnorm(20), ncol=5) ## 4 samples with 5 features
## Find the distances between each sample pair to cluster them
d <- dist(dataset, method = "euclidean", upper=TRUE, diag=TRUE)
tree <- hclust(d)
## Height cutoff.
hlim <- max(tree$height)
## Find clusters.
clusters <- tool.cluster.static(tree, hlim)
```

---

tool.coalesce *Calculate overlaps between groups (main function)*

Description

tool.coalesce is utilized to merge and trim either overlapping modules (containing shared genes) or overlapping genes (containing shared markers)

Usage

```
tool.coalesce(items, groups, rcutoff = 0, ncore = NULL)
```

Arguments

items	array of item identities
groups	array of group identities for items
rcutoff	maximum overlap not coalesced
ncore	minimum number of items required for trimming

Value

a data list with the following components:

CLUSTER	cluster identities after merging and triming (a subset of group identities)
ITEM	item identities
GROUPS	comma separated overlapping group identities

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

Examples

```
## read the coexpr module file as an example:
moddata <- tool.read(system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics"))

## let us find the overlapping ratio between first 10 modules in the file:
## to merge overlapping modules first collect member genes:
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]

## Merge and trim overlapping modules.(max allowed overlap ratio is 0.33)
rmax <- 0.33
moddata$OVERLAP <- moddata$MODULE
moddata <- tool.coalesce(items=moddata$GENE, groups=moddata$MODULE,
rcutoff=rmax)
moddata$MODULE <- moddata$CLUSTER
moddata$GENE <- moddata$ITEM
moddata$OVERLAP <- moddata$GROUPS
moddata <- moddata[,c("MODULE", "GENE", "OVERLAP")]
moddata <- unique(moddata)
```

`tool.coalesce.exec` *Find, merge, and trim overlapping clusters*

Description

`tool.coalesce.exec` searchs overlaps, iteratively merges and trims overlapping clusters (by using `tool.coalesce.find` and `tool.coalesce.merge`, respectively) until no more overlap is available, and assigns representative label for the merged clusters.

Usage

```
tool.coalesce.exec(items, groups, rcutoff, ncore)
```

Arguments

`items`	array of item identities
`groups`	array of group identities for items
`rcutoff`	maximum overlap not coalesced
`ncore`	minimum number of items required for trimming

Value

a data list with the following components:

`CLUSTER`	cluster identities after merging and triming (a subset of group identities)
`GROUPS`	comma separated overlapping group identities

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

Examples

```
## Generate item and group labels for 100 items:
## Assume that unique gene number (items) is 60:
members <- 1:100 ## will be updated
modules <- 1:100 ## will be updated
set.seed(1)
for (i in 1:10){
## each time pick 10 items (genes) from 60 unique item labels
members[(i*10-9):(i*10)] <- sample(60,10)
}
```

```
## Assume that unique group labels is 30:
for (i in 1:10){
## each time pick 10 items (genes) from 30 unique group labels
modules[(i*10-9):(i*10)] <- sample(30, 10)
}
rcutoff <- 0.33
ncore <- length(members)
## Find and trim clusters after iteratively merging the overlapping ones:
res <- tool.coalesce.exec(members, modules, rcutoff, ncore)
```

`tool.coalesce.find` *Find overlapping clusters*

Description

`tool.coalesce.find` finds overlapped clusters of the given `data` according to a given overlapping ratio by using `tool.overlap` and `tool.cluster`, respectively.

Usage

```
tool.coalesce.find(data, rmax)
```

Arguments

`data`	a list including ITEM identities and their GROUP identities
`rmax`	maximum overlap not coalesced

Value

data list including clustering results and following components:

`CLUSTER`	cluster label
`NODE`	item (node) name

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

Examples

```
## Generate item and group labels for 100 items:
## Assume that unique gene number (items) is 60:
members <- 1:100 ## will be updated
modules <- 1:100 ## will be updated
set.seed(1)
for (i in 1:10){
## each time pick 10 items (genes) from 60 unique item labels
members[(i*10-9):(i*10)] <- sample(60,10)
}
## Assume that unique group labels is 30:
for (i in 1:10){
## each time pick 10 items (genes) from 30 unique group labels
modules[(i*10-9):(i*10)] <- sample(30, 10)
}
rcutoff <- 0.33
ncore <- length(members)
## Default output.
res <- data.frame(CLUSTER=modules, GROUPS=modules, ITEM=members,
stringsAsFactors=FALSE)
## Iterative merging and trimming.
res$COUNT <- 0.0
while(TRUE) {
clust <- tool.coalesce.find(res, rcutoff)
if(is.null(clust)) break
res <- tool.coalesce.merge(clust, ncore)
}
```

---

tool.coalesce.merge *Merge overlapping clusters*

Description

tool.coalesce.merge determines combinable groups and trims clusters by removing rarest items.

Usage

```
tool.coalesce.merge(data, ncore)
```

Arguments

data	data list including following components: CLUSTER: cluster label NODE: item (node) name
ncore	minimum number of items required for trimming

Value

res	data list including GROUPS, ITEMs, and their hit COUNTs

Author(s)

Ville-Petteri Makinen

References

Shu L, Zhao Y, Kurt Z, Byars SG, Tukiainen T, Kettunen J, Orozco LD, Pellegrini M, Lusis AJ, Ripatti S, Zhang B, Inouye M, Makinen V-P, Yang X. Mergeomics: multidimensional data integration to identify pathogenic perturbations to biological systems. BMC genomics. 2016;17(1):874.

Examples

```
## Generate item and group labels for 100 items:
## Assume that unique gene number (items) is 60:
members <- 1:100 ## will be updated
modules <- 1:100 ## will be updated
set.seed(1)
for (i in 1:10){
## each time pick 10 items (genes) from 60 unique item labels
members[(i*10-9):(i*10)] <- sample(60,10)
}
## Assume that unique group labels is 30:
for (i in 1:10){
## each time pick 10 items (genes) from 30 unique group labels
modules[(i*10-9):(i*10)] <- sample(30, 10)
}
rcutoff <- 0.33
ncore <- length(members)
## Default output.
res <- data.frame(CLUSTER=modules, GROUPS=modules, ITEM=members,
stringsAsFactors=FALSE)
## Iterative merging and trimming.
res$COUNT <- 0.0
while(TRUE) {
clust <- tool.coalesce.find(res, rcutoff)
if(is.null(clust)) break
res <- tool.coalesce.merge(clust, ncore)
}
```

---

tool.fdr *Estimate False Discovery Rates (FDR)*

Description

`tool.fdr` estimates FDRs for modules as another module statistic.

Usage

```
tool.fdr(p, f = NULL)
```

Arguments

p	p-values of modules
f	pre-defined threshold for FDR

Details

FDRs of modules can be obtained by using either empirical method or Benjamini and Hochberg method.

Value

res	data list including the estimated false discovery rates of modules

Author(s)

Ville-Petteri Makinen

See Also

tool.fdr.empirical, tool.fdr.bh

Examples

```
## let us assume we have a set of pvalues
## and would like to find FDR values:
set.seed(1)
p <- abs(rnorm(10))*1e-2
FDRs <- tool.fdr(p) ## default method is Benjamini Hochberg
```

---

tool.fdr.bh *Benjamini and Hochberg False Discovery Rate*

Description

tool.fdr.bh estimates FDRs of modules by using Benjamini and Hochberg method.

Usage

```
tool.fdr.bh(p)
```

Arguments

p	p-values of modules

Value

res	data list including the estimated false discovery rates of modules

Author(s)

Ville-Petteri Makinen

See Also

tool.fdr, tool.fdr.bh

Examples

```
## let us assume we have a set of pvalues
## and would like to find FDR values:
set.seed(1)
p <- abs(rnorm(10))*1e-2
FDRs <- tool.fdr.bh(p) ## the default method is already Benjamini Hochberg
```

---

tool.fdr.empirical *Estimate Empirical False Discovery Rates*

Description

tool.fdr.empirical estimates empirical FDR for modules

Usage

```
tool.fdr.empirical(p, f0)
```

Arguments

p	p-values of modules
f0	pre-defined threshold for FDR

Value

res	data list including the estimated false discovery rates of modules

Author(s)

Ville-Petteri Makinen

See Also

tool.fdr, tool.fdr.bh

Examples

```
## let us assume we have a set of pvalues
## and would like to find FDR values:
set.seed(1)
p <- abs(rnorm(10))*1e-2
f = 0.05 ## pre-defined threshold for FDR
FDRs <- tool.fdr.empirical(p, f)
```

`tool.graph` *Convert an edge list to a graph representation*

Description

`tool.graph` translates an edge list including TAIL, HEAD and WEIGHT information into a graph representation-adapted data list. It also provides in-degree and out-degree statistics for nodes.

Usage

```
tool.graph(edges)
```

Arguments

`edges`	a data frame with three columns TAIL, HEAD and WEIGHT

Value

a datalist including following components:

`nodes`	N-element array of node names
`tails`	K-element array of node indices
`heads`	K-element array of node indices
`weights`	K-element array of edge weights
`tail2edge`	N-element list of adjacent edge indices
`head2edge`	N-element list of adjacent edge indices
`outstats`	N-row data frame of out-degree node statistics
`instats`	N-row data frame of in-degree node statistics
`stats`	N-row data frame of node statistics

Author(s)

Ville-Petteri Makinen

See Also

`tool.subgraph`

Examples

```
job.kda <- list()
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## module file:
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1
job.kda$nperm <- 20 # the default value is 2000, use 20 for unit tests

## kda.start() process takes long time while seeking hubs in the given net
## Here, we used a very small subset of the module list (1st 10 mods
## from the original module file):
moddata <- tool.read(job.kda$modfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
## save this to a temporary file and set its path as new job.kda$modfile:
tool.save(moddata, "subsetof.supersets.txt")
job.kda$modfile <- "subsetof.supersets.txt"

job.kda <- kda.configure(job.kda)
## Import data for weighted key driver analysis:
## Import topology.
edgdata <- kda.start.edges(job.kda)
## Create an indexed graph structure.
job.kda$graph <- tool.graph(edgdata)

## Remove the temporary files used for the test:
file.remove("subsetof.supersets.txt")
```

`tool.graph.degree` *Find degrees of the nodes*

Description

`tool.graph.degree` finds in-degree and out-degree statistics of the network by using edge lists of the nodes. It also obtains the strenghts of the degrees by using edge weights.

Usage

```
tool.graph.degree(node2edge, weights)
```

Arguments

node2edge	edge list of each node
weights	strengths of the edges

Details

Degree of a node means number of the neighbors belonging to that node. Hence, out-degree statistics are applicable for tail nodes; while in-degree statistics are applicable for the heads.

Value

res	a data list including degree and its strength for each node

Author(s)

Ville-Petteri Makinen

See Also

tool.graph

Examples

```
job.kda <- list()
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## module file:
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1
job.kda$nperm <- 20 # the default value is 2000, use 20 for unit tests

## kda.start() process takes long time while seeking hubs in the given net
## Here, we used a very small subset of the module list (1st 10 mods
## from the original module file):
moddata <- tool.read(job.kda$modfile)
```

```
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
## save this to a temporary file and set its path as new job.kda$modfile:
tool.save(moddata, "subsetof.supersets.txt")
job.kda$modfile <- "subsetof.supersets.txt"

job.kda <- kda.configure(job.kda)

## Import data for weighted key driver analysis:
## Import topology.
edges <- kda.start.edges(job.kda)
## Create an indexed graph structure.
tails <- as.character(edges$TAIL)
heads <- as.character(edges$HEAD)
wdata <- as.double(edges$WEIGHT)

nedges <- length(tails)
# Create factorized representation.
labels <- as.character(c(tails, heads))
labels <- as.factor(labels)
labelsT <- as.integer(labels[1:nedges])
labelsH <- as.integer(labels[(nedges+1):(2*nedges)])
# Create edge lists.
nodnames <- levels(labels)
nnodes <- length(nodnames)
elistT <- tool.graph.list(labelsT, nnodes)
elistH <- tool.graph.list(labelsH, nnodes)
## Collect edge degree stats:
res <- list()
res$nodes <- as.character(nodnames)
res$outstats <- tool.graph.degree(elistT, wdata) ## out degrees
res$instats <- tool.graph.degree(elistH, wdata)  ## in degrees
res$stats <- (res$outstats + res$instats)

## Remove the temporary files used for the test:
file.remove("subsetof.supersets.txt")
```

`tool.graph.list` *Return edge list for each node*

Description

`tool.graph.list` finds and returns the edge list of each node for both tail and head node lists.

Usage

```
tool.graph.list(entries, nnodes)
```

Arguments

entries	either tail nodes list or head nodes list
nnodes	total number of all nodes including both tails and heads

Value

groups	a data list including edge list of each node

Author(s)

Ville-Petteri Makinen

See Also

tool.graph

Examples

```
job.kda <- list()
job.kda$label<-"HDLC"
## parent folder for results
job.kda$folder<-"Results"
## Input a network
## columns: TAIL HEAD WEIGHT
job.kda$netfile<-system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics")
## module file:
job.kda$modfile<- system.file("extdata","mergedModules.txt",
package="Mergeomics")
## "0" means we do not consider edge weights while 1 is opposite.
job.kda$edgefactor<-0.0
## The searching depth for the KDA
job.kda$depth<-1
## 0 means we do not consider the directions of the regulatory interactions
## while 1 is opposite.
job.kda$direction <- 1
job.kda$nperm <- 20 # the default value is 2000, use 20 for unit tests

## kda.start() process takes long time while seeking hubs in the given net
## Here, we used a very small subset of the module list (1st 10 mods
## from the original module file):
moddata <- tool.read(job.kda$modfile)
mod.names <- unique(moddata$MODULE)[1:min(length(unique(moddata$MODULE)),
10)]
moddata <- moddata[which(!is.na(match(moddata$MODULE, mod.names))),]
## save this to a temporary file and set its path as new job.kda$modfile:
tool.save(moddata, "subsetof.supersets.txt")
job.kda$modfile <- "subsetof.supersets.txt"

job.kda <- kda.configure(job.kda)
```

```
## Import data for weighted key driver analysis:
## Import topology.
edges <- kda.start.edges(job.kda)
## Create an indexed graph structure.
tails <- as.character(edges$TAIL)
heads <- as.character(edges$HEAD)
wdata <- as.double(edges$WEIGHT)

nedges <- length(tails)
# Create factorized representation.
labels <- as.character(c(tails, heads))
labels <- as.factor(labels)
labelsT <- as.integer(labels[1:nedges])
labelsH <- as.integer(labels[(nedges+1):(2*nedges)])
# Create edge lists.
nodnames <- levels(labels)
nnodes <- length(nodnames)
elistT <- tool.graph.list(labelsT, nnodes)
elistH <- tool.graph.list(labelsH, nnodes)
## Remove the temporary files used for the test:
file.remove("subsetof.supersets.txt")
```

`tool.metap` *Estimate meta P-values*

Description

`tool.metap` returns the meta p-values of given datasets with multiple p-values.

Usage

```
tool.metap(datasets, idcolumn, pcolumn, weights = NULL)
```

Arguments

`datasets`	data list, whose meta p-values will be obtained
`idcolumn`	column number of the datasets that includes identities
`pcolumn`	column number of the datasets that includes p-values
`weights`	weight list of the data list

Value

`res`	data list including identities and meta p-values of the given `datasets`

Author(s)

Ville-Petteri Makinen

Examples

```
set.seed(1)
## let us assume we have p-values for the coexpr modules obtained from
## distinct analyses by using different gene-marker mapping sets (e.g. eQTLs
## from diff tissues) and we would like to make a meta-analysis for
## these multiple Pvalues of the modules:
datasets=list()
## we have 3 datasets and 3 diff result sets
datasets[[1]] <- data.frame(MODULE=c("Mod1", "Mod2", "Mod3", "Mod4"),
P=c(rnorm(4)))
datasets[[2]] <- data.frame(MODULE=c("Mod1", "Mod2", "Mod3", "Mod4"),
P=c(rnorm(4)))
datasets[[3]] <- data.frame(MODULE=c("Mod1", "Mod2", "Mod3", "Mod4"),
P=c(rnorm(4)))
idcolumn <- "MODULE" ## identifiers of the modules are in the 1st col
pcolumn <- "P" ## p values of the modules are in the 2nd col
tool.metap(datasets, idcolumn, pcolumn)
```

---

tool.normalize *Estimate statistical scores based on Gauss distribution*

Description

To estimate the both pre-liminary and final p-values, `tool.normalize` normalizes the given data, `x`, based on Gaussian distribution defined by `prm` if it is provided. If `prm` is not provided `tool.normalize` utilizes the mean and std dev of `x`.

Usage

```
tool.normalize(x, prm = NULL, inverse = FALSE)
```

Arguments

`x`	data that is aimed to be normalized and produced by a simulation process
`prm`	normalization will take place according to the specified Gaussian distribution parameters, i.e. mean and std dev. If it is not specified, Gaussian statistics of `x` will be obtained and utilized
`inverse`	specifies whether the normalization takes place in reverse order

Value

`prm`	transformed (normalized) parameters for either enrichment score or p-values

Author(s)

Ville-Petteri Makinen

Examples

```
set.seed(1)
## let us assume we have a set of simulated enrichment scores and
## one observed score
x <- rnorm(10) ## obtained from 1st permutation test
obs <- rnorm(1)
## Estimate preliminary P-value:
param <- tool.normalize(x)
z <- tool.normalize(obs, param)
p <- pnorm(z, lower.tail=FALSE)

## Estimate final P-value.
y <- rnorm(10) ## obtained from 2nd permutation test
param <- tool.normalize(c(x, y))
z <- tool.normalize(obs, param)
p <- pnorm(z, lower.tail=FALSE)
p <- max(p, .Machine$double.xmin)
```

---

tool.normalize.quality

Check normalization quality

Description

`tool.normalize.quality` checks transformation quality by using Kolmogorov-Smirnov Test. It seeks the best log transform parameter within the previously specified upper and lower limits, and applies the log transform with the best log parameter.

Usage

```
tool.normalize.quality(g, z)
```

Arguments

`g`	normalization quality control will take place according to the normal distribution parameters defined by `g`, e.g. it can be normal distribution with 0-mean and std dev 1.
`z`	transformed data, i.e. either p-value or enrichment score

Value

`res`	statitics of Kolmogorov-Smirnov Test result obtained for `z` values

Author(s)

Ville-Petteri Makinen

See Also

tool.normalize

Examples

```
set.seed(1)
## let us assume we have a set of normalized scores:
z <- abs(rnorm(10)) ## it should be positive and at least 10 length-vector
z <- z/median(z)
## Find the best log transform.
gamma <- optim(par=1.0, fn=tool.normalize.quality, gr=NULL, z,
lower=-9, upper=9, control=list(reltol=1e-3))
## After finding the best log transform, apply transform:
z <- log(exp(gamma$par)*z + 1.0)
```

---

tool.overlap *Calculate overlaps between groups of specified items*

Description

tool.overlap checks each pair of blocks, finds number of shared items, and obtains significance values of the sharings for block pairs.

Usage

```
tool.overlap(items, groups, nbackground = NULL)
```

Arguments

items	array of item identities
groups	array of group identities for items
nbackground	total number of items

Value

a data list including following components

A	group name
B	group name
POSa	group name rank
POSb	group name rank
Na	group A size
Nb	group B size
Nab	shared items
R	overlap ratio
F	fold change to null expectation
P	overlap P-value (Fisher's test)

Author(s)

Ville-Petteri Makinen

Examples

```
## read the coexpr module file as an example:
moddata <- tool.read(system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics"))
## let us find the overlapping ratio between two modules:
## pick the first and last modules:
mod.names <- unique(moddata$MODULE)[c(1,length(unique(moddata$MODULE)))]
if(length(mod.names) > 0){
modA.members <- moddata[which(moddata$MODULE == mod.names[1]),]
modB.members <- moddata[which(moddata$MODULE == mod.names[2]),]
}
mod.pool <- rbind(modA.members, modB.members)
overlap.stats <- tool.overlap(mod.pool[,2], mod.pool[,1])
```

---

tool.read *Read a data frame from a file*

Description

`tool.read` reads contents of given input file.

Usage

```
tool.read(file, vars = NULL)
```

Arguments

`file`	file name to be read
`vars`	if we want to read particular attributes (columns) from the input file, we need to specify names of these attributes within list `vars` (attribute names can be e.g. MODULE, GENE, LOCUS, etc.)

Details

All lines with NAs are excluded.

Value

`dat`	data frame including content of the given file. If `vars` is specified, only the listed columns inside the `vars` list will be returned.

Author(s)

Ville-Petteri Makinen

Examples

```
## read the network file as an example:
net.info <- tool.read(system.file("extdata","network.mouseliver.mouse.txt",
package="Mergeomics"))
dim(net.info)
names(net.info)
```

`tool.save` *Save a data frame in tab-delimited file*

Description

`tool.save` saves a given data frame into a specified file within a given directory.

Usage

```
tool.save(frame, file, directory = NULL, verbose = TRUE, compression = FALSE)
```

Arguments

`frame`	data frame to be saved into file
`file`	name of the output file to be written
`directory`	path of the directory for the file
`verbose`	specifies whether the information about file saving process will be displayed to user
`compression`	specifies whether the file is compressed while saving. Applicable for only UNIX-family systems with gzip.

Value

`fname`	returns file name with full path

Note

Compression only works on UNIX-family systems with gzip.

Author(s)

Ville-Petteri Makinen

Examples

```
aa<- data.frame(MODULE=c("Mod1", "Mod1", "Mod2", "Mod2", "Mod3"),
NODE=c("GeneA", "GeneC", "GeneB", "GeneC", "GeneA"))
tool.save(aa, "aa.save.txt")
file.remove("aa.save.txt") ## delete the saved file!
```

`tool.subgraph` *Determine network neighbors for a set of nodes*

Description

`tool.subgraph` finds the sub-network, i.e. neighborhood, for a given seed node list with a specified depth. It also provides graph statistics (degrees and strengths) for seed nodes.

Usage

```
tool.subgraph(graph, seeds, depth = 1, direction = 0)
```

Arguments

`graph`	a datalist including following components:
	`nodes: N-element array of node names`
	`tails: K-element array of node indices`
	`heads: K-element array of node indices`
	`weights: K-element array of edge weights`
	`tail2edge: N-element list of adjacent edge indices`
	`head2edge: N-element list of adjacent edge indices`
	`outstats: N-row data frame of out-degree node statistics`
	`instats: N-row data frame of in-degree node statistics`
	`stats: N-row data frame of node statistics`

`seeds`	list of seed node names
`depth`	the maximum number of links to connect neighbors
`direction`	sets the directionality: use a negative value for dowstream, positive for upstream or zero for undirected

Value

a data list including following components:

`RANK`	indices of neighboring nodes (including seeds)
`LEVEL`	number of edges away from seed
`STRENG`	sum of adjacent edge weights within neighborhood
`DEGREE`	number of adjacent edges within neighborhood

Author(s)

Ville-Petteri Makinen

See Also

`tool.subgraph.search`

Examples

```
data(job_kda_analyze)
## take the first node in the graph as the seed, find its neighborhood:
center.node = job.kda$graph$nodes[1]
subnet = tool.subgraph(job.kda$graph, center.node, depth=1, direction=0)
```

`tool.subgraph.find` *Find edges to adjacent nodes*

Description

`tool.subgraph.find` finds the edge lists between given seed nodes and their neighbors

Usage

```
tool.subgraph.find(seeds, edgemap, heads, visited)
```

Arguments

`seeds`	seed nodes' indices
`edgemap`	list of adjacent edge information for entire graph. `edgemap` can belong to either tails or heads.
`heads`	list of either head (destination) or tail (source) nodes of the entire graph
`visited`	flag holding already visited node indices during neighborhood searching

Value

`neighbors`	neighbor edge lists of seed nodes (for either tails or heads)

Note

Neighbor edge lists of the seed nodes should be obtained separately for tail and head nodes.

Author(s)

Ville-Petteri Makinen

Examples

```
data(job_kda_analyze)
depth <- 1
direction <- 0
## Take one or multiple center nodes (seeds) to search the neighborhoods:
## e.g. take the first node in the graph as the seed, find its neighborhood:
center.node = job.kda$graph$nodes[1]
## Convert center node (seed) names to indices:
nodes <- job.kda$graph$nodes
ranks <- match(center.node, nodes)
```

```
ranks <- ranks[which(ranks > 0)]
## we already know that rank is 1, since we took the first node in the graph
## as an example:
ranks <- as.integer(ranks)
## Find edges to adjacent nodes. (both up- and down-stream searches)
visited <- ranks
foundT <- tool.subgraph.find(ranks, job.kda$graph$tail2edge,
job.kda$graph$heads, visited)
foundH <- tool.subgraph.find(ranks, job.kda$graph$head2edge,
job.kda$graph$tails, visited)
```

`tool.subgraph.search` *Search neighborhoods for given nodes*

Description

`tool.subgraph.search` looks for both upstream and downstream neighborhoods of given seed node list for a given depth, gets the directed edge information among seed nodes and their neighbors, obtains statistics (degrees and strengths) for seed nodes.

Usage

```
tool.subgraph.search(graph, seeds, depth, direction)
```

Arguments

`graph`	a datalist including following components:
	`nodes: N-element array of node names`
	`tails: K-element array of node indices`
	`heads: K-element array of node indices`
	`weights:  K-element array of edge weights`
	`tail2edge: N-element list of adjacent edge indices`
	`head2edge: N-element list of adjacent edge indices`
	`outstats: N-row data frame of out-degree node statistics`
	`instats: N-row data frame of in-degree node statistics`
	`stats: N-row data frame of node statistics`

`seeds`	seed nodes' indices
`depth`	the maximum number of links to connect neighbors
`direction`	sets the directionality: use a negative value for dowstream, positive for upstream or zero for undirected

Value

a data list including seed nodes neighborhood information with following components:

`RANK`	indices of neighboring nodes (including seeds)
`LEVEL`	number of edges away from seed
`STRENG`	sum of adjacent edge weights within neighborhood
`DEGREE`	number of adjacent edges within neighborhood

Author(s)

Ville-Petteri Makinen

Examples

```
data(job_kda_analyze)
depth <- 1
direction <- 0
## Take one or multiple center nodes (seeds) to search the neighborhoods:
## e.g. take the first node in the graph as the seed, find its neighborhood:
center.node = job.kda$graph$nodes[1]
## Convert center node (seed) names to indices:
nodes <- job.kda$graph$nodes
ranks <- match(center.node, nodes)
ranks <- ranks[which(ranks > 0)]
## we already know that rank is 1, since we took the first node in the graph
## as an example:
ranks <- as.integer(ranks)
## Find neighbors.
res <- tool.subgraph.search(job.kda$graph, ranks, depth, direction)
```

---

tool.subgraph.stats *Calculate node degrees and strengths*

Description

tool.subgraph.stats graph statistics (degrees and strengths) of the seed nodes obtained from their neighborhoods.

Usage

```
tool.subgraph.stats(frame, edgemap, heads, weights)
```

Arguments

frame	a data frame including following components: RANK: indices of neighboring nodes (including seeds) LEVEL: number of edges away from seed STRENG: sum of adjacent edge weights within neighborhood DEGREE: number of adjacent edges within neighborhood
edgemap	list of adjacent edge information for detected neighborhoods of seed nodes. edgemap can belong to either tails or heads.
heads	list of either head (destination) or tail (source) nodes for neighborhoods of the seed nodes
weights	weights of the edges in the entire graph

Value

a data list including seed nodes neighborhood information with following components:

RANK	indices of neighboring nodes (including seeds)
LEVEL	number of edges away from seed
STRENG	sum of adjacent edge weights within neighborhood
DEGREE	number of adjacent edges within neighborhood

Author(s)

Ville-Petteri Makinen

Examples

```
data(job_kda_analyze)
depth <- 1
direction <- 0
## Take one or multiple center nodes (seeds) to search the neighborhoods:
## e.g. take the first node in the graph as the seed, find its neighborhood:
center.node = job.kda$graph$nodes[1]
## Convert center node (seed) names to indices:
nodes <- job.kda$graph$nodes
ranks <- match(center.node, nodes)
ranks <- ranks[which(ranks > 0)]
## we already know that rank is 1, since we took the first node in the graph
## as an example:
ranks <- as.integer(ranks)
## Find edges to adjacent nodes. (both up- and down-stream searches)
visited <- ranks
levels <- 0*ranks
for(i in 1:depth) {
## Find edges to adjacent nodes.
foundT <- tool.subgraph.find(ranks, job.kda$graph$tail2edge,
job.kda$graph$heads, visited)
foundH <- tool.subgraph.find(ranks, job.kda$graph$head2edge,
job.kda$graph$tails, visited)
## Expand neighborhood for the further depths of the neighborhood search
ranks <- unique(c(foundT, foundH))
visited <- c(visited, ranks)
levels <- c(levels, (0*ranks + i)) ## level shows the depth
if(length(ranks) < 1) break
}
## Calculate node degrees and strengths.
res <- data.frame(RANK=visited, LEVEL=levels, DEGREE=0,
STRENG=0.0, stringsAsFactors=FALSE)
res <- tool.subgraph.stats(res, job.kda$graph$tail2edge,
job.kda$graph$heads, job.kda$graph$weights)
res <- tool.subgraph.stats(res, job.kda$graph$head2edge,
job.kda$graph$tails, job.kda$graph$weights)
```

`tool.translate` *Translate gene symbols*

Description

`tool.translate` converts the symbols given in the list `from` into the list `to`. e.g. we can translate human gene symbols into the mouse orthologs (or vice versa) if the symbol mapping file is provided.

Usage

```
tool.translate(words, from, to)
```

Arguments

`words`	translation table including words (i.e. gene symbols) that will be translated
`from`	a list denoting the words will be translated from which symbols
`to`	a list denoting the words will be translated to which symbols

Value

`words`	translated table (words)

Author(s)

Ville-Petteri Makinen

Examples

```
syms <- tool.read(system.file("extdata", "symbols.txt",
package="Mergeomics"))
syms <- syms[,c("HUMAN", "MOUSE")]
names(syms) <- c("FROM", "TO")
moddata <- tool.read(system.file("extdata",
"modules.mousecoexpr.liver.human.txt", package="Mergeomics"))
moddata$NODE <- moddata$GENE
moddata$NODE <- tool.translate(words=moddata$NODE, from=syms$FROM,
to=syms$TO)
```

`tool.unify` *Convert a distribution to uniform ranks*

Description

`tool.unify` converts a distribution to uniform ranks with respect to a background distribution (or self if no background available).

Usage

```
tool.unify(xtrait, xnull = NULL)
```

Arguments

`xtrait`	the distribution that will be standardized, i.e. uniformly distributed
`xnull`	background distribution to be used to distribute `xtrait` uniformly. If `xnull` is not specified, `xtrait` will be used as background distr

Value

`y`	uniformly distributed form of `xtrait`

Author(s)

Ville-Petteri Makinen

Examples

```
x <- rnorm(10)
y <- tool.unify(x) ## uniformly distributed form of x when null dist is x
z <- tool.unify(x, y) ## uniformly distributed form of x when null dist is y
```

Index

∗ **Integrative Genomics; Multidimensional Data Integration; Gene Networks;**
Mergeomics-package, 3
∗ **Key Drivers**
Mergeomics-package, 3
∗ **datasets**
job.kda, 4
∗
Mergeomics-package, 3

job.kda, 4

kda.analyze, *5*, 5, *8*, *10*, *12*, *15*, *23*, *25*, *27, 28*, *30–32*, *34*, *44*, *54, 55*, *95*
kda.analyze.exec, *6, 7*, 7, *10*, *12*
kda.analyze.simulate, *6*, *8, 9*, 9, *12*
kda.analyze.test, *6*, *8*, *10*, 11, *12*
kda.configure, *14*, 14
kda.finish, 16, *18–20*, *22*, *28*, *30–32*, *34*, *43, 44*
kda.finish.estimate, *16*, 17, *19, 20*, *22*
kda.finish.save, *16*, *18*, 18, *20*, *22*
kda.finish.summarize, *16*, *18–20*, 20, *22*
kda.finish.trim, *16*, *18–21*, 21
kda.prepare, *22*, 22, *23*, *25*, *27, 28*, *30–32*
kda.prepare.overlap, *23, 24*, 24
kda.prepare.screen, *23*, *26*, 26
kda.start, *28*, 28, *30–32*
kda.start.edges, *28, 29*, 29, *32*
kda.start.identify, *28*, *31*, 31
kda.start.modules, *28*, *32*, 32
kda2cytoscape, 33, *34*, *36–39*, *41*, *43*
kda2cytoscape.colorize, 35
kda2cytoscape.colormap, 36
kda2cytoscape.drivers, 37
kda2cytoscape.edges, 39
kda2cytoscape.exec, *39*, 40
kda2cytoscape.identify, 42
kda2himmeli, 43, *44*, *46, 47*, *49*, *51*, *53*
kda2himmeli.colorize, 45
kda2himmeli.colormap, 46
kda2himmeli.drivers, 47
kda2himmeli.edges, 48
kda2himmeli.exec, *48*, 50
kda2himmeli.identify, 52

Mergeomics *(*Mergeomics-package*)*, 3
Mergeomics-package, 3
MSEA.KDA.onestep, *53*, 53, *54*

ssea.analyze, *54, 55*, 55, *56*, *58*, *60*, *63*, *66*, *68*, *71*, *81*, *87*, *95*
ssea.analyze.observe, *57*, 57
ssea.analyze.randgenes, *59*, 59
ssea.analyze.randloci, *62*, 62
ssea.analyze.simulate, *65*, 65
ssea.analyze.statistic, *67*, 67
ssea.control, *56*, *68*, 68, *71*, *81*, *87*
ssea.finish, *56*, 70, *73*, *75*, *77*, *81*, *87*
ssea.finish.details, 72
ssea.finish.fdr, 74
ssea.finish.genes, 76
ssea.meta, 78
ssea.prepare, *56*, *71*, 80, *83*, *85*, *87*
ssea.prepare.counts, 82
ssea.prepare.structure, 84
ssea.start, *56*, *71*, *81*, 86, *90–92*
ssea.start.configure, 88
ssea.start.identify, *91*, 91
ssea.start.relabel, 92
ssea2kda, *56*, *71*, *81*, *87*, 94, *97*, *100*
ssea2kda.analyze, 97
ssea2kda.import, 99

tool.aggregate, 101
tool.cluster, 102, *107*
tool.cluster.static, 103
tool.coalesce, 104
tool.coalesce.exec, 106
tool.coalesce.find, *106*, 107

tool.coalesce.merge, *106*, 108
tool.fdr, 109, *111*
tool.fdr.bh, *110*, 110, *111*
tool.fdr.empirical, *110*, 111
tool.graph, 112, *114*, *116*
tool.graph.degree, 113
tool.graph.list, 115
tool.metap, 117
tool.normalize, 118, *120*
tool.normalize.quality, 119
tool.overlap, *107*, 120
tool.read, 121
tool.save, 122
tool.subgraph, *112*, 123
tool.subgraph.find, 124
tool.subgraph.search, *123*, 125
tool.subgraph.stats, 126
tool.translate, 128
tool.unify, 129