Package 'ccmap'

January 9, 2025

Type Package

Title Combination Connectivity Mapping

Version 1.32.0

Author Alex Pickering

Maintainer Alex Pickering <alexvpickering@gmail.com>

Description Finds drugs and drug combinations that are predicted to reverse or mimic gene expression signatures. These drugs might reverse diseases or mimic healthy lifestyles.

License MIT + file LICENSE

LazyData TRUE

RoxygenNote 6.1.1

Encoding UTF-8

VignetteBuilder knitr

Suggests crossmeta, knitr, rmarkdown, testthat, lydata

Imports AnnotationDbi (>= 1.36.2), BiocManager (>= 1.30.4), ccdata (>= 1.1.2), doParallel (>= 1.0.10), data.table (>= 1.10.4), foreach (>= 1.4.3), parallel (>= 3.3.3), xgboost (>= 0.6.4), lsa (>= 0.73.1)

biocViews GeneExpression, Transcription, Microarray, DifferentialExpression

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

git_branch RELEASE_3_20

git_last_commit eb9f22a

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2025-01-09

Contents

get_dprimes *Extract unbiased effect sizes from meta-analysis by crossmeta.*

Description

Function extracts mu (overall mean effect size) and dprimes (unbiased effect sizes from each contrast).

Usage

```
get_dprimes(es)
```

Arguments

es	Result of call to es_meta.

Details

Result used to query connectivity map drugs and predicted drug combinations.

Value

List containing:

meta	Named numeric vector with overall mean effect sizes for all genes from meta-analysis.
contrasts	List of named numeric vectors (one per contrast) with unbiased effect sizes for all measured genes.

See Also

es_meta.

Examples

```
library(crossmeta)
library(lydata)

data_dir <- system.file("extdata", package = "lydata")

# gather GSE names
gse_names  <- c("GSE9601", "GSE15069", "GSE50841", "GSE34817", "GSE29689")

# load previous differential expression analysis
anals <- load_diff(gse_names, data_dir)

# run meta-analysis
es <- es_meta(anals)

#get dprimes
dprimes <- get_dprimes(es)
```

query_combos *Get overlap between query and predicted drug combination signatures.*

Description

Drugs with the largest positive and negative cosine similarity are predicted to, respectively, mimic and reverse the query signature. Values range from +1 to -1.

Usage

```
query_combos(query_genes, drug_info = c("cmap", "l1000"),
  method = c("average", "ml"), include = NULL,
  ncores = parallel::detectCores())
```

Arguments

query_genes	Named numeric vector of differentual expression values for query genes. Usually 'meta' slot of `get_dprimes` result.
drug_info	Character vector specifying which dataset to query (either 'cmap' or 'l1000'). Can also provide a matrix of differential expression values for drugs or drug combinations (rows are genes, columns are drugs).
method	One of 'average' (default) or 'ml' (machine learning - see details and vignette).
include	Character vector of drug names for which combinations with all other drugs will be predicted and queried. If `NULL` (default), all two drug combinations will be predicted and queried.
ncores	Integer, number of cores to use for method 'average'. Default is to use all cores.

Details

To predict and query all 856086 two-drug cmap combinations, the 'average' `method` can take as little as 10 minutes (Intel Core i7-6700). The 'ml' (machine learning) `method` takes two hours on the same hardware and requires ~10GB of RAM but is slightly more accurate. Both methods will run faster by specifying only a subset of drugs using the `include` parameter. To speed up the 'ml' method, the MRO+MKL distribution of R can help substantially (link). The combinations of LINCS l1000 signatures (~26 billion) can also be queried using the 'average' `method`. In order to compare l1000 results to those obtained with cmap, only the same genes should be queried (see example).

Value

Vector of cosine similarities between query and drug combination signatures.

Examples

```
library(lydata)
library(crossmeta)

# location of data
data_dir <- system.file("extdata", package = "lydata")
```

```
# gather GSE names
gse_names  <- c("GSE9601", "GSE15069", "GSE50841", "GSE34817", "GSE29689")

# load previous analysis
anals <- load_diff(gse_names, data_dir)

# perform meta-analysis
es <- es_meta(anals)

# get dprimes
dprimes <- get_dprimes(es)

# query combinations of metformin and all other cmap drugs
top_met_combos <- query_combos(dprimes$all$meta, include = 'metformin', ncores = 1)

# previous query but with machine learning method
# top_met_combos <- query_combos(dprimes$all$meta, method = 'ml', include = 'metformin')

# query all cmap drug combinations
# top_combos <- query_combos(dprimes$all$meta)

# query all cmap drug combinations with machine learning method
# top_combos <- query_combos(dprimes$all$meta, method = 'ml')

# query l1000 and cmap using same genes
# library(ccdata)
# data(cmap_es)
# data(l1000_es)
# cmap_es <- cmap_es[row.names(l1000_es), ]

# met_cmap  <- query_combos(dprimes$all$meta, cmap_es,  include = 'metformin')
# met_l1000 <- query_combos(dprimes$all$meta, l1000_es, include = 'metformin')
```

`query_drugs` *Get correlation between query and drug signatures. Determines the pearson correlation between the query and each drug signature. Drugs with the largest positive and negative pearson correlation are predicted to, respectively, mimic and reverse the query signature. Values range from +1 to -1.*

Description

The 230829 LINCS l1000 signatures (drugs & genetic over/under expression) can also be queried. In order to compare l1000 results to those obtained with cmap, only the same genes should be included (see second example).

Usage

```
query_drugs(query_genes, drug_info = c("cmap", "l1000"), sorted = TRUE,
  ngenes = 200, path = NULL)
```

Arguments

query_genes	Named numeric vector of differentual expression values for query genes. Usually 'meta' slot of get_dprimes result.
drug_info	Character vector specifying which dataset to query (either 'cmap' or 'l1000'). Can also provide a matrix of differential expression values for drugs or drug combinations (rows are genes, columns are drugs).
sorted	Would you like the results sorted by decreasing similarity? Default is TRUE.
ngenes	The number of top differentially-regulated (up and down) query genes to use if path is NULL. If path is not NULL, ngenes is the larger of 15 or the number of pathway genes with absolute dprimes > 2.
path	Character vector specifying KEGG pathway. Used to find drugs that most closely mimic or reverse query signature for specific pathway.

Value

Vector of pearson correlations between query and drug combination signatures.

See Also

query_combos to get similarity between query and predicted drug combination signatures. diff_path and path_meta to perform pathway meta-analysis.

Examples

```
# Example 1 -----

library(crossmeta)
library(ccdata)
library(lydata)

data_dir <- system.file("extdata", package = "lydata")
data(cmap_es)

# gather GSE names
gse_names  <- c("GSE9601", "GSE15069", "GSE50841", "GSE34817", "GSE29689")

# load previous differential expression analysis
anals <- load_diff(gse_names, data_dir)

# run meta-analysis
es <- es_meta(anals)

# get meta-analysis effect size values
dprimes <- get_dprimes(es)

# most significant pathway (from path_meta)
path <- 'Amino sugar and nucleotide sugar metabolism'

# query using entire transcriptional profile
topd <- query_drugs(dprimes$all$meta, cmap_es)

# query restricted to transcriptional profile for above pathway
topd_path <- query_drugs(dprimes$all$meta, cmap_es, path=path)
```

```
# Example 2 -----

# create drug signatures
genes <- paste("GENE", 1:1000, sep = "_")
set.seed(0)

drug_info <- data.frame(row.names = genes,
                        drug1  = rnorm(1000, sd = 2),
                        drug2  = rnorm(1000, sd = 2),
                        drug3  = rnorm(1000, sd = 2))

# query signature is drug3
query_sig <- drug_info$drug3
names(query_sig) <- genes

res <- query_drugs(query_sig, as.matrix(drug_info))

# use only common genes for l1000 and cmap matrices
# library(ccdata)
# data(cmap_es)
# data(l1000_es)
# cmap_es <- cmap_es[row.names(l1000_es), ]
```

`sum_rowcolCumsum` *Sum of cumulative sum computed over rows then columns of matrix.*

Description

Equivalent to computing the cumulative sum of a matrix over rows, then over columns, then suming every value (though much faster and more memory efficient).

Usage

```
sum_rowcolCumsum(x, i, j)
```

Arguments

`x`	Numeric vector of non-zero values of matrix.
`i`	Integer vector of row indices of x.
`j`	Integer vector of column indices of x.

Value

Numeric value equal to the sum of the cumulative sum computed over rows then columns of a matrix.

Examples

```
x <- c(1, 1, 1, -1) # non-zero values of matrix
i <- c(1, 2, 3, 4)  # row indices of x
j <- c(4, 1, 3, 2)  # col indices of x

sum_rowcolCumsum(x, i, j)
```

Index