Package ‘ccmap’

January 14, 2017

**Type** Package

**Title** Combination Connectivity Mapping

**Version** 1.0.0

**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** 5.0.1

**VignetteBuilder** knitr

**Suggests** crossmeta, knitr, rmarkdown, testthat, lydata,

**Imports** AnnotationDbi (>= 1.34.4), BiocInstaller, ccdata (>= 0.99.4), doParallel (>= 1.0.10), data.table (>= 1.9.6), foreach (>= 1.4.3), parallel (>= 3.3.1), xgboost (>= 0.4.4)

**biocViews** GeneExpression, Transcription, Microarray, DifferentialExpression

**NeedsCompilation** no

---

**R topics documented:**

- get_dprimes ........................................... 2
- query_combos ......................................... 3
- query_drugs ........................................... 4
- sum_rowcolCumsum .................................... 5

**Index** 6
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 net overlap are predicted to, respectively, mimic and reverse the query signature. A value of 1 would indicate that all drug and query genes are regulated in the same direction and have the same order when sorted by absolute changes in differential expression. A value of -1 would indicate that all drug and query genes are regulated in the opposite direction and have the same order when sorted by absolute changes in differential expression.

Usage

query_combos(query_genes, method = "average", include = NULL, ncores = parallel::detectCores())

Arguments

- **query_genes**: Named numeric vector of differential expression values for query genes. Usually `meta` slot of `get_dprimes` result.
- **method**: One of `average` (default) or `ml` (machine learning - see details and vignette).
- **include**: Character vector of cmap drug names for which combinations with all other cmap drugs will be predicted and queried. If `NULL` (default), all 856086 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 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).

Value

Vector of numeric values between 1 and -1 indicating extent of overlap between query and drug combination signatures (see description).

Examples

```r
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
```
query_drugs

Get overlap between query and drug signatures.

Description
Determines the volume under the surface formed by plotting net overlap (z) as a function of number of drug and query genes (x and y).

Usage
query_drugs(query_genes, drug_info = NULL, sorted = TRUE)

Arguments
query_genes  Named numeric vector of differential expression values for query genes. Usually 'meta' slot of get_dprimes result.
drug_info  Matrix of differential expression values for drugs or drug combinations. Rows are genes, columns are drugs.
sorted  Would you like the results sorted in decreasing order of overlap? Default is TRUE.

Details
Drugs with the largest positive and negative net overlap are predicted to, respectively, mimic and reverse the query signature. A value of 1 would indicate that all drug and query genes are regulated in the same direction and have the same order when sorted by absolute changes in differential expression. A value of -1 would indicate that all drug and query genes are regulated in the opposite direction and have the same order when sorted by absolute changes in differential expression.

Value
Vector of numeric values between 1 and -1 indicating extent of overlap between query and drug signatures (see description).
sum_rowcolCumsum

See Also

query_combos to get overlap between query and predicted drug combination signatures.

Examples

```r
# 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))
```

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

```r
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

```r
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

es_meta, 2
get_dprimes, 2
query_combos, 3, 5
query_drugs, 4
sum_rowcolCumsum, 5