Package ‘mslp’

May 30, 2024

Type Package

Title Predict synthetic lethal partners of tumour mutations

Version 1.6.0

Description An integrated pipeline to predict the potential synthetic lethality partners (SLPs) of tumour mutations, based on gene expression, mutation profiling and cell line genetic screens data. It has built-in support for data from cBioPortal. The primary SLPs correlating with mutations in WT and compensating for the loss of function of mutations are predicted by random forest based methods (GENIE3) and Rank Products, respectively. Genetic screens are employed to identify consensus SLPs leads to reduced cell viability when perturbed.

License GPL-3

Encoding UTF-8

LazyData false

RoxygenNote 7.2.3

Depends R (>= 4.2.0)

Imports data.table (>= 1.13.0), doRNG, fmsb, foreach, magrittr, org.Hs.eg.db, pROC, randomForest, RankProd, stats, utils

Suggests BiocStyle, doFuture, future, knitr, rmarkdown, roxygen2, tinytest

VignetteBuilder knitr

biocViews Pharmacogenetics, Pharmacogenomics

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

git_branch RELEASE_3_19

git_last_commit 23e640d

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-29

Author Chunxuan Shao [aut, cre]

Maintainer Chunxuan Shao <chunxuan@outlook.com>
## Description

Identify SLPs compensating for the loss of function of mutations. The up-regulated SLPs are selected via the rank products algorithm, with option calculateProduct = FALSE for a robust results and capacity on large datasets.

## Usage

```r
comp_slp(
    zscore_data,
    mut_data,
    mutgene = NULL,
    positive_perc = 0.5,
    p_thresh = 0.01,
    ...
)
```

## Arguments

- **zscore_data**: a matrix (genes by patients) reflecting gene expression related to wide type samples. For example, generated from `pp_tcga`.
- **mut_data**: a data.table with columns "patientid" and "mut_entrez".
- **mutgene**: identify SLPs for specific mutation (gene symbols). If NULL (by default), the intersection genes between zscore_data and mut_data are used.
**cons_slp**

Identify consensus SLPs

**Description**

Identify consensus SLPs based on Cohen’s Kappa or hypergeometric test.

**Usage**

```r
cons_slp(screen_slp, tumour_slp)
```

**Arguments**

- `screen_slp`: screen hits data annotated with SLPs information, generated by `scr_slp`.
- `tumour_slp`: the merged SLPs data predicted by `corr_slp` and `comp_slp`.

**positive_perc**

Keep genes with positive z-score in at least `positive_perc` * number of mutation patients.

**p_thresh**

P-value threshold to filter out results.

... Additional parameters to `RankProducts`.

**Value**

A data.table with predicted SLPs.

- `mut_entrez`: Entrez ids of mutations.
- `mut_symbol`: Gene symbols of mutations.
- `slp_entrez`: Entrez ids of SLPs.
- `slp_symbol`: Gene symbols of SLPs.
- `pvalue`: P-value from `RankProducts`.
- `fdr`: "BH" adjusted p-value via `p.adjust`.

**Examples**

```r
#- Toy examples, see vignette for more.
#- Add the parallel backend.
require(future)
require(doFuture)
plan(multisession, workers = 2)
data("example_z")
data("example_comp_mut")
res <- comp_slp(example_z, example_comp_mut)
plan(sequential)
```
Details

Consensus SLPs are enriched screen hits that are SLPs of same mutations in different cell lines. For each common mutation, the SLPs predicted from human tumour data are used as the total sets. We used either Cohen’s Kappa coefficient on a confusion matrix, or Hypergeometric test, to test the significance of overlapping of screen hits.

Value

A data.table.

mut_entrez  Entrez ids of mutations.
mut_symbol  Gene symbols of mutations.
cons_slp_entrez  Entrez ids of consensus SLPs.
cons_slp_symbol  Gene symbols of Consensus SLPs.
cell_1, cell_2  From which pair of cell lines the consensus SLPs predicted.
judgement  Judgement based on Cohen’s Kappa.
kappa_value  Cohen’s Kappa coefficient
pvalue  pvalue for Cohen’s Kappa coefficient.
fdr  "BH" adjusted pvalue via p.adjust.

References


Examples

#- See the examples in the vignette.
if (FALSE) k_res <- cons_slp(scr_res, merged_res)

**corr_slp**  
Identify SLPs via correlation

Description

Identify SLPs of mutations based on co-expression. GENIE3 is employed to find genes highly correlated with mutations in wide type patients.

Usage

corr_slp(
  expr_data,
  mut_data,
  mutgene = NULL,
  im_thresh = 0.001,
  topgene = 2000,
  ...
)
est_im

Arguments

- `expr_data`: an expression matrix, genes by patients.
- `mut_data`: a data.table with columns "patientid" and "mut_entrez".
- `mutgene`: identify SLPs for specific mutation (gene symbols). If NULL (by default), the intersection genes between expr_data and mut_data are used.
- `im_thresh`: minimum importance threshold.
- `topgene`: top N genes above the `im_thresh`.
- `...`: further parameters to `genie3`.

Value

A data.table with predicted SLPs.

- `mut_entrez`: Entrez ids of mutations.
- `mut_symbol`: Gene symbols of mutations.
- `slp_entrez`: Entrez ids of SLPs.
- `slp_symbol`: Gene symbols of SLPs.
- `fdr`: "BH" adjusted pvalue via `p.adjust`.
- `im`: The importance value returned by `genie3`.

Examples

```r
# Toy examples, see vignette for more.
require(future)
require(doFuture)
plan(multisession, workers = 2)
data("example_expr")
data("example_corr_mut")
res <- corr_slp(example_expr, example_corr_mut)
plan(sequential)
```

Description

Estimate the importance threshold based on repetition GENIE3 results via ROC.

Usage

```r
est_im(permu_data, fdr_thresh = 0.001)
```

Arguments

- `permu_data`: permuted `corr_slp` results.
- `fdr_thresh`: fdr threshold to selected "TRUE" SLPs.
Details

We first generate a SLPs by repetition matrix from repetition GENIE3 results. SLPs with high im value in repetitions are selected and considered as "TRUE" SLPs via the rank product algorithm. Then for each repetition, we perform receiver operating characteristic curve analysis and select an optimal threshold by "youden" approach. The optimal thresholds are averaged to get the final threshold.

Value

A data.table with mut_entrez (mutation entrez_id) and roc_thresh (estimated im threshold).

Examples

```r
# Toy examples.
require(future)
require(doFuture)
plan(multisession, workers = 2)
data(example_expr)
data(example_corr_mut)
mutgene <- sample(intersect(example_corr_mut$mut_entrez, rownames(example_expr)), 2)
nperm <- 5
res <- lapply(seq_len(nperm), function(x) corr_slp(example_expr, example_corr_mut, mutgene = mutgene))
roc_thresh <- est_im(res)
plan(sequential)
```

---

data(example_compSLP)  

SLPs predicted by comp_slp

Description

SLPs predicted by comp_slp

Usage

data(example_compSLP)

Format

A data.table.
### example_comp_mut

Patients mutations to be use in the comp_slp

**Description**

Mutations and related TCGA ids.

**Usage**

```r
data(example_comp_mut)
```

**Format**

A data.table.

---

### example_corrSLP

SLPs predicted by corr_slp

**Description**

SLPs predicted by corr_slp

**Usage**

```r
data(example_corrSLP)
```

**Format**

A data.table.

---

### example_corr_mut

Patients mutations to be use in the corr_slp

**Description**

Mutations and related TCGA ids.

**Usage**

```r
data(example_corr_mut)
```

**Format**

A data.table.
example_expr  
Expression data to be used in comp_slp

Description
    Expression matrix, genes by samples.

Usage
    data(example_expr)

Format
    A matrix.

example_z  
Expression data to be used in corr_slp

Description
    Z score matrix, genes by samples.

Usage
    data(example_z)

Format
    A matrix.

genie3  
Run GENIE3

Description
    Calculate the weight matrix between genes via randomForest, modified from original codes by Huynh-Thu, V.A.
Usage

genie3(
  expr.matrix,
  ngene = NULL,
  K = "sqrt",
  nb.trees = 1000,
  input.idx = NULL,
  importance.measure = "IncNodePurity",
  trace = FALSE,
  ...
)

Arguments

desp.matrix  expression matrix (genes by samples).
ngene  an integer, only up to the first ngene (included) targets (responsible variables).
K  choice of number of input genes randomly, must be one of "sqrt", "all", an integer.
nb.trees  number of trees in ensemble for each target gene (default 1000).
input.idx  subset of genes used as input genes (default all genes). A vector of indices or
gene names is accepted.
importance.measure  type of variable importance measure, "IncNodePurity" or "%IncMSE".
trace  index of currently computed gene is reported (default FALSE).
...  parameter to randomForest.

Value

A weighted adjacency matrix of inferred network, element w_ij (row i, column j) gives the impor-
tance of the link from regulatory gene i to target gene j.

References

from Expression Data Using Tree-Based Methods. PLoS ONE 5, e12776.

Examples

#- Toy examples.
mtx <- matrix(sample(1000, 100), nrow = 5)
mtx <- rbind(mtx[1, ] * 2 + rnorm(20), mtx)
colnames(mtx) <- paste0("s_", seq_len(ncol(mtx)))
rownames(mtx) <- paste0("g_", seq_len(nrow(mtx)))
res <- genie3(mtx, nb.trees = 100)
getlink  

Get sorted list of regulatory links in GENIE3 results

**Description**

Take genie3 output and sort the links.

**Usage**

getlink(weight.matrix, report.max = NULL)

**Arguments**

weight.matrix  
a weighted adjacency matrix as returned by genie3.

report.max  
maximum number of links to report (default all links).

**Value**

A data.table of links with columns "from.gene", "to.gene", "im".

**Examples**

```r
mtx <- matrix(sample(1000, 100), nrow = 5)
mtx <- rbind(mtx[1, ] * 2 + rnorm(20), mtx)
colnames(mtx) <- paste0("s_", seq_len(ncol(mtx)))
rownames(mtx) <- paste0("g_", seq_len(nrow(mtx)))
res <- genie3(mtx, nb.trees = 10)
res_link <- getlink(res)
```

merge_slp  

Merge SLPs

**Description**

Merge predicted SLPs from comp_slp and corr_slp.

**Usage**

merge_slp(comp_data, corr_data)

**Arguments**

comp_data  
predicted SLPs from `comp_slp`.

corr_data  
predicted SLPs from `corr_slp`. 
Value

A data.table.

- **mut_entrez**: Entrez ids of mutations.
- **mut_symbol**: Gene symbols of mutations.
- **slp_entrez**: Entrez ids of SLPs.
- **slp_symbol**: Gene symbols of SLPs.
- **pvalue**: p_value from `RankProducts`.
- **fdr**: "BH" adjusted pvalue via `p.adjust`.
- **im**: The importance value returned by `genie3`.
- **dualhit**: Whether the slp is identified by `corr_slp` and `comp_slp`.

Examples

```r
data("example_z")
data("example_comp_mut")
comp_res <- comp_slp(example_z, example_comp_mut)

data("example_expr")
data("example_corr_mut")
corr_res <- corr_slp(example_expr, example_corr_mut)

res <- merge_slp(comp_res, corr_res)
```

---

`pp_tcga`  
*Process tumour genomic data*

Description

Preprocess mutation, cna, expression and zscore datsets in TCGA PanCancer Atlas by cBioPortal.

Usage

```r
pp_tcga(
  p_mut,  
p_cna,  
p_exprs,  
p_score,  
freq_thresh = 0.02,  
expr_thresh = 10,  
hypermut_thresh = 300
)
```
Arguments

- `p_mut`: path of mutation data, like "data_mutations_uniprot.txt" provided by cBioPortal.
- `p_cna`: path of copy number variation data, like "data_CNA.txt".
- `p_exprs`: path of normalized RNAseq expression data, like "data_RNA_Seq_v2_expression_median.txt".
- `p_score`: path of zscore data, like "data_RNA_Seq_v2_mRNA_median_Zscores.txt".
- `freq_thresh`: threshold to select recurrent mutations.
- `expr_thresh`: threshold to remove low expression genes.
- `hypermut_thresh`: threshold for hpyermutations.

Details

It is designed to process the TCGA data provided by cBioPortal. In mutation data, "Missense_Mutation", "Nonsense_Mutation", "Frame_Shift_Del", "Frame_Shift_Ins", "In_Frame_Del", "In_Frame_Ins", "Nonstop_Mutation" are selected for the downstream analysis. In CNA data, genes with GISTIC value equal to -2 are used. Patients with hypermutations are removed. Low expression genes, or genes that are not detected in any patient are filtered out.

Value

Return a list of mut_data, expr_data and zscore_data, while expr_data and zscore_data are matrix (entrez_id by patients), mut_data is a data.table with two columns of "patientid" and "mut_entrez".

References


Examples

```r
#- See vignette for more details.
if (FALSE) {
  P_mut <- "data_mutations_extended.txt"
  P_cna <- "data_CNA.txt"
  P_expr <- "data_RNA_Seq_v2_expression_median.txt"
  P_z   <- "data_RNA_Seq_v2_mRNA_median_Zscores.txt"
  res   <- pp_tcga(P_mut, P_cna, P_expr, P_z)
  saveRDS(res$mut_data, "mut_data.rds")
  saveRDS(res$expr_data, "expr_data.rds")
  saveRDS(res$zscore_data, "zscore_data.rds")
}
```
Identify SLPs in screen hits

Description
Identify whether screen hits are SLPs of mutations detected in both patients and cell lines, based on predicted SLPs in `corr_slp` and `comp_slp`.

Usage
`scr_slp(cell, screen_data, cell_mut, tumour_slp)`

Arguments
- `cell`: a cell line.
- `screen_data`: a data.table of genomic screen results with three columns, "screen_entrez", "screen_symbol" and "cell_line".
- `cell_mut`: cell line mutation data.
- `tumour_slp`: merged SLPs.

Value
A data.table.

- `cell_line`: Name of cell lines.
- `screen_entrez`: Entrez ids of hits.
- `screen_symbol`: Gene symbols of hits.
- `mut_entrez`: Entrez ids of mutations.
- `mut_symbol`: Gene symbols of mutations.
- `is_slp`: Whether the targeted gene is a SLP.
- `pvalue`: p.value from `RankProducts`.
- `fdr`: "BH" adjusted pvalue via `p.adjust`.
- `im`: The importance value returned by `genie3`.
- `dualhit`: Whether the slp is identified by `corr_slp` and `comp_slp`.

Examples
```r
require(future)
require(doFuture)
plan(multisession, workers = 2)
library(magrittr)
library(data.table)
data(example_compSLP)
data(example_corrSLP)
merged_res <- merge_slp(example_compSLP, example_corrSLP)
```
# Toy hits data.
screen_1 <- merged_res[, .(slp_entrez, slp_symbol)] %>%
  unique %>%
  .[sample(nrow(.), round(.8 * nrow(.)))] %>%
  setnames(c(1, 2), c("screen_entrez", "screen_symbol")) %>%
  .[, cell_line := "cell_1"]

screen_2 <- merged_res[, .(slp_entrez, slp_symbol)] %>%
  unique %>%
  .[sample(nrow(.), round(.8 * nrow(.)))] %>%
  setnames(c(1, 2), c("screen_entrez", "screen_symbol")) %>%
  .[, cell_line := "cell_2"]

screen_hit <- rbind(screen_1, screen_2)

# Toy mutations data.
mut_1 <- merged_res[, .(mut_entrez)] %>%
  unique %>%
  .[sample(nrow(.), round(.8 * nrow(.)))] %>%
  .[, cell_line := "cell_1"]

mut_2 <- merged_res[, .(mut_entrez)] %>%
  unique %>%
  .[sample(nrow(.), round(.8 * nrow(.)))] %>%
  .[, cell_line := "cell_2"]

mut_info <- rbind(mut_1, mut_2)

# Hits that are identified as SLPs.
scr_res <- lapply(c("cell_1", "cell_2"), scr_slp, screen_hit, mut_info, merged_res)
scr_res[lengths(scr_res) == 0] <- NULL
scr_res <- rbindlist(scr_res)
plan(sequential)
Index

* datasets
  example_comp_mut, 7
  example_compSLP, 6
  example_corr_mut, 7
  example_corrSLP, 7
  example_expr, 8
  example_z, 8
  comp_slp, 2, 3, 10, 11, 13
  cons_slp, 3
  corr_slp, 3, 4, 5, 10, 11, 13
  est_im, 5
  example_comp_mut, 7
  example_compSLP, 6
  example_corr_mut, 7
  example_corrSLP, 7
  example_expr, 8
  example_z, 8
  genie3, 5, 8, 11, 13
  getlink, 10
  merge_slp, 10
  p.adjust, 3–5, 11, 13
  pp_tcga, 2, 11
  RankProducts, 3, 11, 13
  scr_slp, 3, 13