

# Package ‘cTRAP’

April 15, 2019

**Title** Identification of candidate causal perturbations from differential gene expression data

**Version** 1.0.3

**Description** Compare differential gene expression results with those from known cellular perturbations (such as gene knock-down, overexpression or small molecules) derived from the Connectivity Map. Such analyses allow not only to infer the molecular causes of the observed difference in gene expression but also to identify small molecules that could drive or revert specific transcriptomic alterations.

**Depends** R (>= 3.5.0)

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**biocViews** DifferentialExpression, GeneExpression, RNASeq, Transcriptomics, Pathways, ImmunoOncology

**URL** <https://github.com/nuno-agostinho/cTRAP>

**BugReports** <https://github.com/nuno-agostinho/cTRAP/issues>

**Suggests** testthat, knitr, covr, biomaRt

**RoxygenNote** 6.1.1

**Imports** data.table, limma, stats, fgsea, pbapply, plyr, cowplot, ggplot2, rhdf5, R.utils, httr, methods, piano, readr, utils, graphics

**VignetteBuilder** knitr

**git\_url** <https://git.bioconductor.org/packages/cTRAP>

**git\_branch** RELEASE\_3\_8

**git\_last\_commit** e97c3f1

**git\_last\_commit\_date** 2018-12-03

**Date/Publication** 2019-04-15

**Author** Bernardo P. de Almeida [aut],  
Nuno Saraiva-Agostinho [aut, cre],  
Nuno L. Barbosa-Morais [aut, led]

**Maintainer** Nuno Saraiva-Agostinho <nunodanielagostinho@gmail.com>

**R topics documented:**

|   |    |
|---|----|
| compareAgainstL1000 . . . . .             | 2  |
| cTRAP . . . . .                           | 3  |
| downloadENCODEknockdownMetadata . . . . . | 4  |
| downloadENCODExamples . . . . .           | 4  |
| downloadL1000data . . . . .               | 5  |
| filterL1000metadata . . . . .             | 6  |
| getL1000conditions . . . . .              | 6  |
| getL1000perturbationTypes . . . . .       | 7  |
| loadL1000perturbations . . . . .          | 7  |
| performDifferentialExpression . . . . .   | 8  |
| plotL1000comparison . . . . .             | 9  |
| prepareENCODegeneExpression . . . . .     | 10 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>11</b> |
|--------------|-----------|

---

|                     |                                       |
|---------------------|---------------------------------------|
| compareAgainstL1000 | <i>Compare against L1000 datasets</i> |
|---------------------|---------------------------------------|

---

**Description**

Compare against L1000 datasets

**Usage**

```
compareAgainstL1000(diffExprGenes, perturbations, cellLine,
  method = c("spearman", "pearson", "gsea"), geneSize = 150,
  pAdjustMethod = "BH")
```

**Arguments**

|               |   |
|---------------|---|
| diffExprGenes | Numeric: named vector of differentially expressed genes where the name of the vector are gene names and the values are a statistic that represents significance and magnitude of differentially expressed genes (e.g. t-statistics) |
| perturbations | L1000perturbations object: file with L1000 loaded perturbations (check <a href="#">loadL1000perturbation</a> )  |
| cellLine      | Character: cell line(s)   |
| method        | Character: comparison method (spearman, pearson or gsea)  |
| geneSize      | Number: top and bottom differentially expressed genes to use for gene set enrichment (GSE) (only used if method is gsea)  |
| pAdjustMethod | Character: method for p-value adjustment (for more details, see <a href="#">p.adjust.methods</a> ; only used if method is spearman or pearson)  |

**Value**

Data table with correlation or GSEA results comparing differential gene expression values with those associated with L1000 perturbations

## Examples

```
cellLine <- "HepG2"
data("L1000perturbationsSmallMolecules")
perturbations <- L1000perturbationsSmallMolecules
data("diffExprStat")

# Compare against L1000 using Spearman correlation
compareAgainstL1000(diffExprStat, perturbations, cellLine,
                    method="spearman")

# Compare against L1000 using Pearson correlation
compareAgainstL1000(diffExprStat, perturbations, cellLine,
                    method="pearson")

# Compare against L1000 using gene set enrichment analysis (GSEA)
compareAgainstL1000(diffExprStat, perturbations, cellLine, method="gsea")
```

---

cTRAP

*cTRAP package*

---

## Description

Compare differential gene expression results with those from big datasets (e.g. L1000), allowing to infer which types of perturbations may explain the observed difference in gene expression.

## Details

**Input:** To use this package, a named vector of differentially expressed gene metric is needed, where its values represent the significance and magnitude of the differentially expressed genes (e.g. t-statistic) and its names are gene symbols.

**Workflow:** The differentially expressed genes will be compared against selected perturbation conditions by:

- Spearman or Pearson correlation with z-scores of differentially expressed genes after perturbations from L1000. Use function `compareAgainstL1000` with `method = "spearman"` or `method = "pearson"`
- Gene set enrichment analysis (GSEA) using the (around) 12 000 genes from L1000. Use function `compareAgainstL1000` with `method = gsea`.

Available perturbation conditions for L1000 include:

- Cell line(s).
- Perturbation type (gene knockdown, gene upregulation or drug intake).
- Drug concentration.
- Time points.

Values for each perturbation type can be listed with `getL1000perturbationTypes()`

**Output:** The output includes a data frame of ranked perturbations based on the associated statistical values and respective p-values.

downloadENCODeknockdownMetadata

*Download metadata for ENCODE knockdown experiments*

---

**Description**

Download metadata for ENCODE knockdown experiments

**Usage**

```
downloadENCODeknockdownMetadata(cellLine = NULL, gene = NULL)
```

**Arguments**

|          |                        |
|----------|------------------------|
| cellLine | Character: cell line   |
| gene     | Character: target gene |

**Value**

Data frame containing ENCODE knockdown experiment metadata

**Examples**

```
downloadENCODeknockdownMetadata("HepG2", "EIF4G1")
```

---

downloadENCODEsamples *Download ENCODE samples*

---

**Description**

Download ENCODE samples

**Usage**

```
downloadENCODEsamples(metadata)
```

**Arguments**

|          |                            |
|----------|----------------------------|
| metadata | Character: ENCODE metadata |
|----------|----------------------------|

**Value**

List of loaded ENCODE samples

**Examples**

```

if (interactive()) {
  # Download ENCODE metadata for a specific cell line and gene
  cellLine <- "HepG2"
  gene <- "EIF4G1"
  ENCODEmetadata <- downloadENCODEknockdownMetadata(cellLine, gene)

  # Download samples based on filtered ENCODE metadata
  ENCODEsamples <- downloadENCODEsamples(ENCODEmetadata)
}

```

---

|                   |                            |
|-------------------|----------------------------|
| downloadL1000data | <i>Download L1000 data</i> |
|-------------------|----------------------------|

---

**Description**

The data will be downloaded if not available

**Usage**

```

downloadL1000data(file, type = c("metadata", "geneInfo", "zscores"),
  zscoresId = NULL)

```

**Arguments**

|           |  |
|-----------|--|
| file      | Character: filepath  |
| type      | Character: type of data to load  |
| zscoresId | Character: identifiers to partially load z-scores file (for performance reasons) |

**Value**

Metadata as a data table

**Examples**

```

# Download L1000 metadata
l1000metadata <- downloadL1000data("l1000metadata.txt", "metadata")

# Download L1000 gene info
downloadL1000data("l1000geneInfo.txt", "geneInfo")

# Download L1000 zscores based on filtered metadata
l1000metadataKnockdown <- filterL1000metadata(
  l1000metadata, cellLine="HepG2",
  perturbationType="Consensus signature from shRNAs targeting the same gene")

if (interactive()) {
  downloadL1000data("l1000zscores.gctx.gz", "zscores",
    l1000metadataKnockdown$sig_id)
}

```

---

filterL1000metadata     *Filter L1000 metadata*

---

### Description

Filter L1000 metadata

### Usage

```
filterL1000metadata(metadata, cellLine = NULL, timepoint = NULL,
  dosage = NULL, perturbationType = NULL)
```

### Arguments

|                  |  |
|------------------|--|
| metadata         | Data frame: metadata   |
| cellLine         | Character: cell line (if NULL, all values are loaded)                        |
| timepoint        | Character: timepoint (if NULL, all values are loaded)                        |
| dosage           | Character: dosage (if NULL, all values are loaded)                           |
| perturbationType | Character: type of perturbation (if NULL, all perturbation types are loaded) |

### Value

Filtered L1000 metadata

### Examples

```
data("l1000metadata")
# l1000metadata <- downloadL1000data("l1000metadata.txt", "metadata")
filterL1000metadata(l1000metadata, cellLine="HEPG2", timepoint="2 h",
  dosage="25 ng/mL")
```

---

getL1000conditions     *List available conditions in L1000 datasets*

---

### Description

Downloads metadata if not available

### Usage

```
getL1000conditions(metadata, control = FALSE)
```

### Arguments

|          |  |
|----------|--|
| metadata | Data table: L1000 metadata                     |
| control  | Boolean: show controls for perturbation types? |

**Value**

List of conditions in L1000 datasets

**Examples**

```
data("l1000metadata")
# l1000metadata <- downloadL1000data("l1000metadata.txt", "metadata")
getL1000conditions(l1000metadata)
```

---

getL1000perturbationTypes  
*Get perturbation types*

---

**Description**

Get perturbation types

**Usage**

```
getL1000perturbationTypes()
```

**Value**

Perturbation types and respective codes as used by L1000 datasets

**Examples**

```
getL1000perturbationTypes()
```

---

loadL1000perturbations  
*Load L1000 perturbation data*

---

**Description**

Load L1000 perturbation data

**Usage**

```
loadL1000perturbations(metadata, zscores, geneInfo,
  sanitizeCompoundNames = FALSE)
```

**Arguments**

- metadata            Data frame: L1000 Metadata
- zscores            Data frame: GCTX z-scores
- geneInfo           Data frame: L1000 gene info
- sanitizeCompoundNames  
                    Boolean: replace identifiers with compound names

**Value**

Perturbation data from L1000 as a data table

**Examples**

```
if (interactive()) {
  metadata <- downloadL1000data("l1000metadata.txt", "metadata")
  metadata <- filterL1000metadata(metadata, cellLine="HepG2")
  zscores <- downloadL1000data("l1000zscores.gctx", "zscores",
    metadata$sig_id)
  geneInfo <- downloadL1000data("l1000geneInfo.txt", "geneInfo")
  loadL1000perturbations(metadata, zscores, geneInfo)
}
```

---

performDifferentialExpression

*Perform differential gene expression based on ENCODE data*

---

**Description**

Perform differential gene expression based on ENCODE data

**Usage**

```
performDifferentialExpression(counts)
```

**Arguments**

counts            Data frame: gene expression

**Value**

Data frame with differential gene expression results between knockdown and control

**Examples**

```
data("ENCODEsamples")

## Download ENCODE metadata for a specific cell line and gene
# cellLine <- "HepG2"
# gene <- "EIF4G1"
# ENCODEmetadata <- downloadENCODEknockdownMetadata(cellLine, gene)

## Download samples based on filtered ENCODE metadata
# ENCODEsamples <- downloadENCODEsamples(ENCODEmetadata)

counts <- prepareENCODEgeneExpression(ENCODEsamples)

# Remove low coverage (at least 10 counts shared across two samples)
minReads <- 10
minSamples <- 2
filter <- rowSums(counts[ , -c(1, 2)] >= minReads) >= minSamples
counts <- counts[filter, ]
```



```
## Convert ENSEMBL identifier to gene symbol
# library(biomaRt)
# mart <- useDataset("hsapiens_gene_ensembl", useMart("ensembl"))
# genes <- sapply(strsplit(counts$gene_id, "\\."), `[`, 1)
# geneConversion <- getBM(filters="ensembl_gene_id", values=genes, mart=mart,
#                          attributes=c("ensembl_gene_id", "hgnc_symbol"))
# counts$gene_id <- geneConversion$hgnc_symbol[
#   match(genes, geneConversion$ensembl_gene_id)]

## Perform differential gene expression analysis
# diffExpr <- performDifferentialExpression(counts)
```

---

plotL1000comparison    *Plot L1000 data comparison*

---

## Description

Plot L1000 data comparison

## Usage

```
plotL1000comparison(object, perturbationID, topGenes = TRUE)
```

## Arguments

|                |  |
|----------------|--|
| object         | L1000 comparison object  |
| perturbationID | Character: perturbation identifier   |
| topGenes       | Boolean: plot top (topGenes = TRUE) or bottom genes (topGenes = FALSE) in case of plotting gene set enrichment analysis (GSEA) |

## Value

Plot illustrating the comparison with L1000 data

## Examples

```
data("l1000perturbationsKnockdown")
cellLine <- "HepG2"
compareKnockdown <- list()

# Compare against L1000 using Spearman correlation
compareKnockdown$spearman <- compareAgainstL1000(
  diffExprStat, l1000perturbationsKnockdown, cellLine, method="spearman")

# Compare against L1000 using Pearson correlation
compareKnockdown$pearson <- compareAgainstL1000(
  diffExprStat, l1000perturbationsKnockdown, cellLine, method="pearson")

# Compare against L1000 using gene set enrichment analysis (GSEA) with the top
# and bottom 150 genes
compareKnockdown$gsea <- compareAgainstL1000(
  diffExprStat, l1000perturbationsKnockdown, cellLine, method="gsea",
```

```
geneSize=150)

EIF4G1knockdown <- grep("EIF4G1", compareKnockdown$gsea$genes, value=TRUE)
plotL1000comparison(compareKnockdown$spearman, EIF4G1knockdown)
plotL1000comparison(compareKnockdown$pearson, EIF4G1knockdown)
plotL1000comparison(compareKnockdown$gsea, EIF4G1knockdown)
```

---

```
prepareENCODEgeneExpression
```

*Load an ENCODE gene expression data*

---

### Description

Load an ENCODE gene expression data

### Usage

```
prepareENCODEgeneExpression(samples)
```

### Arguments

`samples` List of loaded ENCODE samples

### Value

Data frame containing gene read counts

### Examples

```
data("ENCODEsamples")
## Download ENCODE metadata for a specific cell line and gene
# cellLine <- "HepG2"
# gene <- "EIF4G1"
# ENCODEmetadata <- downloadENCODEknockdownMetadata(cellLine, gene)

## Download samples based on filtered ENCODE metadata
# ENCODEsamples <- downloadENCODEsamples(ENCODEmetadata)

prepareENCODEgeneExpression(ENCODEsamples)
```

# Index

compareAgainstL1000, [2](#)  
cTRAP, [3](#)  
cTRAP-package (cTRAP), [3](#)  
  
downloadENCODEknockdownMetadata, [4](#)  
downloadENCODEsamples, [4](#)  
downloadL1000data, [5](#)  
  
filterL1000metadata, [6](#)  
  
getL1000conditions, [6](#)  
getL1000perturbationTypes, [7](#)  
  
loadL1000perturbations, [2](#), [7](#)  
  
p.adjust.methods, [2](#)  
performDifferentialExpression, [8](#)  
plotL1000comparison, [9](#)  
prepareENCODEgeneExpression, [10](#)