Package 'ADaM'

March 28, 2019

Type Package

Title Adaptive Daisy Model

Version 0.1.0

Author Clare Pacini and Francesco Iorio

Maintainer Francesco Iorio <francesco.iorio@sanger.ac.uk>

Description The ADaM package implements a semi-supervised algorithm for computing a fuzzy-intersection of non-fuzzy sets by adaptively determining the minimal number of sets to which an element should belong in order to be a member of the fuzzy-intersection (the membership threshold). This threshold maximises the deviance from expectation of the cardinality of the resulting fuzzy-intersection, as well as the coverage of predefined elements. This method can be used to identify the minimal number of cell lines from a given tissue in which the inactivation of a gene (for example via CRISPR-Cas9 targeting) should exert a reduction of viabilty (or fitness effect) in order for that gene to be considered a core-fitness essential gene for the tissue under consideration. This method is used to discriminate between core-fitness and context-specific essential genes in a study describing a large scale genome-wide CRISPR-Cas9 pooled drop-out screening (Behan FM & Iorio F & Picco G et al., Prioritisation of cancer therapeutic targets using CRISPR-Cas9 screens. Nature, In press).

License GPL-2

LazyData true

R topics documented:

`ADAM.empiricalOdds` *Empirical odds of number of fitness genes per number of cell lines*

Description

This function calculates log10 odd ratios of observed vs. expected profiles of cumulative number of fitness genes in fixed number of cell lines. Expected values are the mean of those observed across randomised version of the observed binary matrix.

Usage

```
ADAM.empiricalOdds(observedCumSum,simulatedCumSum)
```

Arguments

`observedCumSum`	Observed profile of cumulative sum of numbers of fitness genes in fixed number of cell lines. This is generated by the `ADAM.panessprofile` function.
`simulatedCumSum`	Random profiles of cumulative sum of fitness genes in fixed number of cell lines. This is generated by the function `ADAM.generateNullModel`.

Value

A named vector:

`odds`	log base 10 odd ratios of observed versus expected cumulative sums of number of fitness genes across fixed numbers of cell lines.

Author(s)

C. Pacini & F. Iorio

See Also

ADAM.panessprofile, ADAM.generateNullModel

Examples

```
data(exampleDepMat)
observed<-ADAM.panessprofile(depMat=exampleDepMat)
null_m<-ADAM.generateNullModel(depMat=exampleDepMat)
logOdds <- ADAM.empiricalOdds(observed$CUMsums,null_m$nullCumSUM)

logOdds
```

`ADAM.generateNullModel` *Generate null profile of number of fitness genes across fixed numbers of cell lines and cumulative sums.*

Description

This function randomly perturbs the binary dependency matrix to generate a null distribution of profiles of fitness genes across fixed number of cell lines, and corresponding null distribution of cumulative sums.

Usage

```
ADAM.generateNullModle(depMat,ntrials=100,display=TRUE)
```

Arguments

`depMat`	Binary dependency matrix, rows are genes and columns are samples. 1 in position *[i,j]* indicates that inactivation of the *i*-th gene exerts a significant loss of fitness in the *j*-th sample, 0 otherwise.
`ntrials`	Integer, default =100. How many times to randomly perturb dependency matrix to generate the null distributions.
`display`	Boolean, default is TRUE. Should bar plots of the null profiles be plotted

Details

For a number of trials specified in (`ntrials`) the inputted binary dependency matrix is randomised, keeping its column marginal sums. The profiles of fitness genes across fixed number of cell lines, and corresponding cumulative sums, are returned for each random perturbation.

Value

A list with the following two matrices

`nullProf`	Matrix of number of fitness genes for fixed number of cell lines from. Each rows of matrix corresponds to a random trial.
`nullCumSum`	Matrix of profile of cumulative number of fitness genes in fixed number of cell lines. Each row of matrix is one random trial.

Author(s)

C. Pacini & F. Iorio

Examples

```
data(exampleDepMat)
pprofile <- ADAM.generateNullModel(depMat = exampleDepMat,ntrials=100)
```

`ADAM.panessprofile`	*Calculate profile of number of fitness genes across fixed numbers of cell lines and cumulative sums.*

Description

This function calculates the numbers (and cumulative numbers) of genes whose inactivation exerts a fitness effect in *n* cell lines, varying *n* from 1 to the number of cell lines in the dataset in input.

Usage

```
ADAM.panessprofile(depMat,
                   display=TRUE,
                   main_suffix='fitness genes in at least 1 cell line',
                   xlab='n. dependent cell lines')
```

Arguments

`depMat`	Binary dependency matrix, rows are genes and columns are samples. 1 in position *[i,j]* indicates that inactivation of the *i*-th gene exerts a significant loss of fitness in the *j*-th sample, 0 otherwise.
`display`	Boolean, default is TRUE. Should bar plots of the dependency profiles be plotted
`main_suffix`	If display=TRUE, title suffix to give to plot of number of genes depleted in a give number of cell lines, default is 'genes depleted in at least 1 cell line'
`xlab`	If display=TRUE, label to give to x-axis of the plots, default is 'n. cell lines'

Value

A list with the following two named vectors:

`panessprof`	Number of genes that are depleted for a number of cell lines
`CUMsums`	Cumulative number of genes depleted in at least x cell lines

Author(s)

C. Pacini & F. Iorio

Examples

```
data(exampleDepMat)
pprofile <- ADAM.panessprofile(depMat = exampleDepMat)
```

`ADAM.randomisedepMat` *Binary matrix randomisation preserving column totals*

Description

This function takes in input a matrix and shuffles its entries column wisely. If the matrix is binary then then matrix resulting from this shuffling will have the same column marginal totals of the inpputted one.

Usage

```
ADAM.randomisedepMat(depMat)
```

Arguments

`depMat`	A numerical matrix.

Value

The matrix given in input with entries shuffled column wisely.

Author(s)

C. Pacini & F. Iorio

Examples

```
data(exampleDepMat)
rnd_exampleDepMat<-ADAM.randomisedepMat(exampleDepMat)
```

`ADAM.truePositiveRate` *Profile of True Positive Rates*

Description

This function calculates a profile of True Positive Rates for fitness genes in at least n cell lines, with positive cases from a reference set of essential genes.

Usage

```
ADAM.truePositiveRate(depMat,essentialGeneSet)
```

Arguments

`depMat`	Binary dependency matrix, rows are genes and columns are samples. 1 in position *[i,j]* indicates that inactivation of the *i*-th gene exerts a significant loss of fitness in the *j*-th sample, 0 otherwise.
`essentialGeneSet`	Reference set of predefined essential genes. This is used to define positive cases.

Details

This function calculates true positive rates for fitness genes in at least n cell lines (for each n). First, this function calculates the number of cell lines each gene is a fitness gene. Second, for a given number of cell lines, the set of genes that are fitness genes in at least that number of cell lines is determined. Finally, this set of genes is then compared to the reference set of essential genes to calculate a true positive rate.

Value

A list of the following vectors:

`P`	Vector of number of genes that are depleted for a number of cell lines.
`TP`	Vector of number of genes in sets of P are true positives, i.e. in the essentialGeneSet.
`TPR`	TP divided by number of genes in set essentialGeneSet to give the true positive rate.

Author(s)

C. Pacini & F. Iorio

Examples

```
data(exampleDepMat)

pprofile<-ADAM.panessprofile(depMat=exampleDepMat)

nullmodel<-ADAM.generateNullModel(depMat=exampleDepMat,ntrials = 1000)

data(curated_BAGEL_essential)

EO<-ADAM.empiricalOdds(observedCumSum = pprofile$CUMsums,simulatedCumSum =nullmodel$nullCumSUM )

TPR<-ADAM.truePositiveRate(exampleDepMat,curated_BAGEL_essential)
```

`curated_BAGEL_essential` *Referecence set of core-fitness essential genes*

Description

Reference set of predefined core-fitness essential genes used in [1], derived from [2] and further curated as follows. In order to avoid their status (essential/non-essential) being defined a priori, a set of high-confidence cancer driver genes from [3] were filtered out.

Usage

```
data(curated_BAGEL_essential)
```

Format

A vector of string with a gene symbol in each entry

References

[1] Behan FM & Iorio F & Picco G et al., Prioritisation of cancer therapeutic targets using CRISPR-Cas9 screens. Nature, In press.

[2] Hart T, Chandrashekhar M, Aregger M, Steinhart Z, Brown KR, MacLeod G, et al. High-Resolution CRISPR Screens Reveal Fitness Genes and Genotype-Specific Cancer Liabilities. Cell. 2015;163:1515–26.

[3] Iorio F, Knijnenburg TA, Vis DJ, Bignell GR, Menden MP, Schubert M, et al. A Landscape of Pharmacogenomic Interactions in Cancer. Cell. 2016;166:740–54.

Examples

```
data(curated_BAGEL_essential)
head(curated_BAGEL_essential)
```

`exampleDepMat` *Example dependency matrix data object*

Description

A binary matrix summarising the fitness effect status (1 = cellular fitness reduced upon gene inactivation via CRISPR-Cas9 targeting) of all the genes (at a genome scale) across 32 human colorectal cancer cell lines, derived from [1].

Usage

```
data("exampleDepMat")
```

Format

The format is: num [1:17995, 1:32] 0 0 0 0 0 0 0 0 0 0 ... - attr(*, "dimnames")=List of 2 ..$: chr [1:17995] "A1BG" "A1CF" "A2M" "A2ML1" ... ..$: chr [1:32] "Cell_line_1" "Cell_line_2" "Cell_line_3" "Cell_line_4" ...

References

[1] Behan FM & Iorio F & Picco G et al., *Prioritisation of cancer therapeutic targets using CRISPR-Cas9 screens*. **Nature**, In press.

Examples

```
data(exampleDepMat)
head(exampleDepMat)
```

Index