Package ‘IdMappingAnalysis’

January 14, 2017

Imports boot, mclust, RColorBrewer, Biobase

Maintainer Alex Lisovich <alex.lisovich@gmail.com>, Roger Day
  <day01@pitt.edu>

License GPL-2

Title ID Mapping Analysis

LazyData yes

Type Package

Author Alex Lisovich, Roger Day

Description Identifier mapping performance analysis

Version 1.18.0

Date 2013-05-24

Depends R (>= 2.14), R.oo (>= 1.13.0), rChoiceDialogs

  ‘idMapDiffCounts.R’ ‘IdMappingAnalysis-package.R’
  ‘jointIdMap.plots.R’ ‘jointIdMap.R’
  ‘jointUniquePairs.extensions.R’
  ‘jointUniquePairs.interactive.plots.R’

biocViews Annotation, MultipleComparison

NeedsCompilation no

R topics documented:

Bootstrap .................................................. 2
Cor .......................................................... 3
CorrData ................................................................ 5
DataFilter ..................................................... 6
Display ......................................................... 7
examples ................................................................ 8
expectedUtility .................................................. 9
fit2clusters ..................................................... 10
IdMap ........................................................... 12
## Bootstrap

The Bootstrap class

### Description

Package: IdMappingAnalysis

**Class Bootstrap**

Object

```
|--- Bootstrap
  |--- IdMapBase
```

**Directly known subclasses:**

public static class **Bootstrap**
extends **IdMapBase**

The Bootstrap object encapsulates a data frame containing the unique pairs in the first two columns and the correlation results, sd and bias obtained from the bootstrapping procedure in the next 3 columns. During the object creation, the bootstrapping procedure is applied to each row of the experiment set pairs from the CorrData object optionally applying the Fisher transform to the correlation data.

### Usage

```
Bootstrap(corrData=NULL, Fisher=FALSE, R=200, verbose=FALSE, ...)
```

### Arguments

- **corrData**
  CorrData object on which the correlation related bootstrapping is performed.

- **Fisher**
  If `TRUE`, the Fisher transform of data is performed during bootstrapping. Default is `FALSE`.

- **R**
  The number of bootstrap replicates. Default is 200.

- **verbose**
  if `TRUE` enables diagnostic messages. Default is `FALSE`.

- **...**
  Not used.
Value
A Bootstrap object encapsulating the `data.frame` with following columns:

- **column 1**: the first component (primary IDs) of unique pairs. The column name corresponds to the primary key of a source ID Map
- **column 2**: the second component (secondary IDs) of unique pairs. The column name corresponds to the secondary key of a source ID Map
- **'corr' column**: contains the correlation values obtained from bootstrapping
- **'sd' column**: contains the correlation sd values obtained from bootstrapping
- **'bias' column**: contains the correlation bias values obtained from bootstrapping

Fields and Methods

Methods:

- **plot**: Scatterplot of bootstrapped results: sd vs correlation.

Methods inherited from `IdMapBase`:
- [, aligned, as.data.frame, dim, dimnames, getName, primaryIDs, primaryKey, secondaryKey

Methods inherited from `Object`:
- $, $<-, [], [<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Alex Lisovich, Roger Day

Examples

```r
bootstrap<-Bootstrap(examples$corrData,R=20,verbose=TRUE);
class(bootstrap);
bootstrap[1:10,];
```

---

**Corr**

* The Corr class

---

Description

Package: IdMappingAnalysis

**Class Corr**

```
Object
```
```
```
```
```
```
```
```
Directly known subclasses:

public static class Corr
extends IdMapBase

Create the Corr object by performing correlations on the CorrData object using the correlation algorithm defined by the method argument. The Corr object encapsulates a data.frame containing three columns: the first two are unique pairs and the third is a correlation results with a column name reflecting the correlation method ('pearson', 'spearman' or 'kendall').

Usage

Corr(corrData=NULL, method="pearson", verbose=FALSE, ...)

Arguments

corrData CorrData object on which correlation is performed or a data.frame compliant with the Corr object internal data frame format.

method Correlation method ('pearson', 'spearman' or 'kendall'). Default is 'pearson'.

verbose if TRUE enables diagnostic messages. Default is FALSE.

Fields and Methods

Methods:

ggetData Extract correlation results from the Corr object.

ggetUniquePairs Extract unique pairs from the Corr object.

plot Plot the density distributions for correlation object(s).

Methods inherited from IdMapBase:

[, aligned, as.data.frame, dim, dimnames, getName, primaryIDs, primaryKey, secondaryKey

Methods inherited from Object:

$. $<-, [. [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Alex Lisovich, Roger Day

Examples

corr<-Corr(examples$corrData,method="spearman",verbose=TRUE);
class(corr);
corr[1:10,];
CorrData class

Description

Package: IdMappingAnalysis
Class CorrData

Object

~~| ~~-++CorrData

Directly known subclasses:

public static class CorrData extends Object

CorrData object stores the pair of experiments on which the correlation related processing is performed (MS/MS and mRNA for example) in such a way that two experiments are aligned by experiment names and by the primary keys ensuring the fast correlations. Typically, the primary ID of the ID Map set under consideration is a primary key for a first experiment, and the secondary ID if the ID Map set is a primary key for a second experiment. The alignment of two experiments by primary keys is guaranteed by using the unique pairs object to produce a matching pair of primary keys on which both experiments are ordered. Represented by a list of two elements with names corresponding to the primary and secondary IDs of the unique pairs ('acc' and 'probeset' for example), each element containing a data frame with primary or secondary IDs in the first column while the rest of columns contain the experiment data. The names of the data columns in both data frames are identical and correspond to the sample IDs. The match of sample IDs and an alignment by primary/secondary IDs is ensured by the proper processing during the object creation.

Usage

CorrData(uniquePairs=NULL, expSet1=NULL, expSet2=NULL, verbose=FALSE, ...)

Arguments

uniquePairs  UniquePairs object or a list of such objects on which a single or a list of CorrData objects is constructed.
expSet1  a first ExperimentSet object with primary IDs corresponding (partially intersecting) with the content the first column of UniquePairs (uniquePairsData) object.
expSet2  a second ExperimentSet object with primary IDs corresponding (partially intersecting) with the content the second column of UniquePairs (uniquePairsData) object.
verbose if TRUE enables diagnostic messages. Default is FALSE.
...  Not used.
Fields and Methods

Methods:

- `as.MultiSet`: Convert CorrData object into MultiSet object.
- `getExperimentSet`: Get experiment set data frame for a given modality.
- `getSampleNames`: Get experiment sample names.
- `getUniquePairs`: Extract unique pairs from the CorrData object.
- `interactive.plot`: Draw a scatterplot of experiment data interactively.
- `plot`: Scatterplot of experiment data.
- `primaryKey`: Retrieves a primary key for a given CorrData object.
- `secondaryKey`: Retrieves a secondary key for a given CorrData object.

Methods inherited from Object:

- `$`, `$<-, `[`, `[<-, `as.character`, `attach`, `attachLocally`, `clearCache`, `clearLookupCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `gc`, `getEnvironment`, `getFieldModifier`, `getFieldModifiers`, `getFields`, `getInstanceCreationTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `objectSize`, `print`, `registerFinalizer`, `save`

Author(s)

Alex Lisovich, Roger Day

Examples

```r
corrData<-CorrData(examples$uniquePairs, examples$msmsExperimentSet, examples$mRNAExperimentSet, verbose=TRUE);
class(corrData);
```

---

DataFilter

The DataFilter class

Description

Package: IdMappingAnalysis

Class DataFilter

Object

```r
~~|
~~+--DataFilter
```

Directly known subclasses:

public static class DataFilter
extends Object

Serves as a wrapper for data data filtering functions define as static methods of the DataFilter class.
Display

Usage

DataFilter()

Fields and Methods

Methods:

- `do.apply` Filter experiment using constraints.
- `fisherTransform` Compute the Fisher transform.
- `fisherTransformInverse` Compute the Fisher inverted transform.
- `fisherTransformJacobeian` Compute the Fisher transform Jacobean.
- `logTen` Compute log10 of a numerical vector combined with thresholding on minimum value.
- `minAvgCountConstraint` Perform mean based thresholding of an input vector.
- `minCountConstraint` Perform minimum count based thresholding of an input vector.
- `minCountGroupConstraint` Perform minimum count based thresholding of an input vector subdivided into groups.
- `removeNAseries` Remove NA series from the experiment set.

Methods inherited from Object:

$ $<-$ $.[[<-$ as.character attach attachLocally clearCache clearLookupCache clone detach equals extend finalize gc getEnvironment getFieldModifier getFields getInstantiationTime getStaticInstance hasField hashCode ll load objectSize print registerFinalizer save

Author(s)

Alex Lisovich, Roger Day

Display The Display class

Description

Package: IdMappingAnalysis

Class Display

Object

---
---Display

Directly known subclasses:

public static class Display extends Object

Serves as a wrapper for a set of graphical functions defined as static methods of Display class
Usage
Display()

Fields and Methods
Methods:

- **copy**: Save current plot to the file.
- **create**: Open a new display device.
- **line.loess**: Plot loess transformed data.
- **line.unsorted**: Draw a curve from unsorted points.
- **progressMsg**: Display a progress message.
- **textBoundingBox**: Determine the size of the text bounding box.
- **zoom.pars**: Zoom graphics parameters.

Methods inherited from Object:
$\$, $\leftarrow$, [[, [\leftarrow, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstanceationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Alex Lisovich, Roger Day

examples

IdMappingAnalysis sample data...

Description
IdMappingAnalysis sample data

Value
A list containing the following sample items:

- **mrnaExperimentSet**: MRNA Experiment sample data frame
- **msmsExperimentSet**: MSMS Experiment sample data frame
- **outcomeMap**: outcome sample data frame
- **identDfList**: list of ID Map sample data frames collected from various services
- **jointIdMap**: sample JointIdMap object for exploring the various DBs quantitative identifier mapping performance
- **jointIdMap_corr**: sample JointIdMap object for exploring the various DBs correlation based identifier mapping performance
- **uniquePairs**: sample UniquePairs object
expectedUtility

jointUniquePairs
   sample JointUniquePairs object
corrData
   sample CorrData object suitable for subsequent fast correlations
corr
   sample Corr object encapsulating the correlation results
mixture
   sample Mixture object encapsulating the mixture modeling results
bootstrap
   sample Bootstrap object encapsulating the results of bootstrapping on correlations

Author(s)
Alex Lisovich, Roger Day

---

**expectedUtility**

*Expected utility of an ID mapping, ID filtering, or other bioinformatics data preparation method*

Description

`expectedUtility` calculates mean expected utility and total expected utility across pairs of features from two bioinformatics platforms. It is used to evaluate an ID mapping, ID filtering, or other bioinformatics data preparation method.

Usage

```r
expectedUtility(dataset, label = "", bootModelCorClusters, columnsToRemove = c("Utp", "Lfp", "deltaPlus", "pi1Hat"), Utp, Lfp, deltaPlus, guarantee = 1e-09)
```

Arguments

dataset A data frame or list from a call to `fit2clusters`, the posterior probabilities for each observation, their variance estimates. See Details.

label A text string describing the method being studied, to label the return value. This is handy for using `rbind` to combine results for different methods.

bootModelCorClusters Source for mixture model estimates. If missing, extracted from calling frame.

columnsToRemove Names of columns to remove from return value.

Utp Utility of a true positive.

Lfp Loss of a false positive.

deltaPlus Parameter defined as Pr("+" | "+" or "0")

guarantee Minimum value for posterior probability.
Details

The input dataset should be a dataframe with one row per ID pair, and the following columns:

- **Utp**: Utility of a true positive.
- **Lfp**: Loss of a false positive.
- **postProb**: The posterior probabilities for each observation
- **postProbVar**: The variances of the posterior probabilities, usually estimated from the bootstrap using Boot

Value

A data frame with just one row. The columns are:

- **Utp**: Utility of a true positive.
- **Lfp**: Loss of a false positive.
- **deltaPlus**: Parameter defined as Pr("+" | "+" or "0")
- **deltaZero**: Parameter defined as Pr("0" | "0" or "x")
- **nPairs**: Number of ID pairs selected by the method.
- **pi1Hat**: The estimate of the probability of the high-correlation component; obtained from
- **PrPlus**: Estimated probability that an ID pair is in the "+" group.
- **PrTrue**: Estimated probability that an ID pair is in the "+" or "0" group: PrPlus/deltaPlus
- **PrFalse**: Estimated probability that an ID pair is in the "-" group.
- **Utrue**: The component of expected utility from "true positives": PrTrue * Utp
- **Lfalse**: The (negative) component of expected utility from "false positives": PrFalse * Lfp
- **Eutility1**: The average expected utility per ID pair: Utrue-Lfalse.
- **Eutility**: The total expected utility, summing over ID pairs: nrow(dataset)*Eutility1.

fit2clusters

Flexible two-cluster mixture fit of a numeric vector

Description

fit2clusters uses an ECM algorithm to fit a two-component mixture model. It is more flexible than mclust in some ways, but it only deals with one-dimensional data.

Usage

fit2clusters(Y, Ylabel = "correlation", Ysigsq, piStart = c(0.5, 0.5), VStart = c(0.1, 0.1), psiStart = c(0, 0.1), NinnerLoop = 1, nReps = 500, psi0Constraint, V0Constraint, sameV = FALSE, estimatesOnly = TRUE, plotMe = TRUE, testMe = FALSE, Ntest = 5000, simPsi = c(0, 0.4), simPi = c(2/3, 1/3), simV = c(0.05^2, 0.05^2), simAlpha = 5, simBeta = 400, seed, ...)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>The vector of numbers to fit.</td>
</tr>
<tr>
<td>Ysigsq</td>
<td>The vector of variance estimates for Y.</td>
</tr>
<tr>
<td>Ylabel</td>
<td>Label for the Y axis in a density fit figure.</td>
</tr>
<tr>
<td>piStart</td>
<td>Starting values for the component proportions.</td>
</tr>
<tr>
<td>VStart</td>
<td>Starting values for the component variances.</td>
</tr>
<tr>
<td>psiStart</td>
<td>Starting values for the component means</td>
</tr>
<tr>
<td>NinnerLoop</td>
<td>Number of iterations in the &quot;C&quot; loop of ECM.</td>
</tr>
<tr>
<td>nReps</td>
<td>Upper limit of number of EM steps.</td>
</tr>
<tr>
<td>psi0Constraint</td>
<td>If not missing, a fixed value for the first component mean.</td>
</tr>
<tr>
<td>V0Constraint</td>
<td>If not missing, a fixed value for the first component variance.</td>
</tr>
<tr>
<td>sameV</td>
<td>If TRUE, the components have the same variance.</td>
</tr>
<tr>
<td>estimatesOnly</td>
<td>If TRUE, return only the estimates. Otherwise, returns details per observations, and return the estimates as an attribute.</td>
</tr>
<tr>
<td>plotMe</td>
<td>If TRUE, plot the mixture density and kernel smooth estimates.</td>
</tr>
<tr>
<td>testMe</td>
<td>If TRUE, run a code test.</td>
</tr>
<tr>
<td>Ntest</td>
<td>For testing purposes, the number of replications of simulated data.</td>
</tr>
<tr>
<td>simPsi</td>
<td>For testing purposes, the true means.</td>
</tr>
<tr>
<td>simPi</td>
<td>For testing purposes, the true proportions</td>
</tr>
<tr>
<td>simV</td>
<td>For testing purposes, the true variances.</td>
</tr>
<tr>
<td>simAlpha</td>
<td>For testing purposes, alpha parameter in rgamma for measurement error variance.</td>
</tr>
<tr>
<td>simBeta</td>
<td>For testing purposes, beta parameter in rgamma for measurement error variance.</td>
</tr>
<tr>
<td>seed</td>
<td>For testing purposes, random seed.</td>
</tr>
<tr>
<td>...</td>
<td>Not used; testing roxygen2.</td>
</tr>
</tbody>
</table>

Details

See the document "ECM_algorithm_for_two_clusters.pdf".

Value

If estimatesOnly is TRUE, return only the estimates: Otherwise, return a dataframe of details per observations, and return the estimates as an attribute. The estimates details are:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pi1</td>
<td>The probability of the 2nd mixture component</td>
</tr>
<tr>
<td>psi0</td>
<td>The mean of the first component (psi0Constraint if provided)</td>
</tr>
<tr>
<td>psi1</td>
<td>The mean of the second component</td>
</tr>
<tr>
<td>Var0</td>
<td>The variance of the first component (V0Constraint if provided)</td>
</tr>
<tr>
<td>Var1</td>
<td>The variance of the second component</td>
</tr>
</tbody>
</table>

The observations details are:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>The original observations.</td>
</tr>
<tr>
<td>Ysigsq</td>
<td>The original measurement variances.</td>
</tr>
<tr>
<td>posteriorOdds</td>
<td>Posterior odds of being in component 2 of the mixture.</td>
</tr>
<tr>
<td>postProbVar</td>
<td>Estimated variance of the posterior probability, using the delta method.</td>
</tr>
</tbody>
</table>
Description

Package: IdMappingAnalysis

Class IdMap

Object
~~|--
~~+=--IdMapBase
~~~~~~|--
~~~~~~+=--IdMap

Directly known subclasses:

public static class IdMap extends IdMapBase

IdMap is an object encapsulating a data frame with two columns (Primary ID and Secondary ID) where primaryID is a character string uniquely identifying the ID under consideration (unprot accessions ID or acc, Entrez Gene ID etc) and the Secondary ID is a comma separated list of secondary IDs associated with a given primary ID for a particular DB service. The analysis typically starts from obtaining a set of ID Maps (from the various DB services) which are not assumed to have the same number of rows or the same set of primary IDs. The process of alignment of this ID Maps is performed within the JointIdMap

Usage

IdMap(DF=NULL, name=null, primaryKey=colnames(DF)[1], secondaryKey=colnames(DF)[2], ...)

Arguments

DF A data.frame consisting of two columns (primary and secondary IDs) from which the IdMap object is to be created.
name A character string representing the name of the given IdMap object. Default is 
primaryKey The name of the primary (first) column in an ID Map. If missing then the input data frame first column name is used and if it is not available defaults to "From".
secondaryKey The name of secondary (second) column in an ID Map. If missing then the input data frame second column name is used and if it is not available defaults to "To".
...

Fields and Methods

Methods:
as

`as.UniquePairs` - Create a UniquePairs object from a given IdMap object.

`as.list` - Coerce an object or a list of compatible object.

`getCounts` - Compute the count of secondary IDs for each primary ID.

`merge` - Merge the IdMap object with a second IdMap object or a list of IdMap objects.

`swapKeys` - Swap the primary and secondary key columns.

Methods inherited from `IdMapBase`:

- ., aligned, as.data.frame, dim, dimnames, getName, primaryIDs, primaryKey, secondaryKey

Methods inherited from `Object`:

- $, $<-, $[,..., as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstanceationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Alex Lisovich, Roger Day

Examples

```r
obj <- IdMap(examples$identDfList[[2]]);
obj$primaryKey();
obj$secondaryKey();
```

---

### Description

Package: IdMappingAnalysis

**Class `IdMapBase`**

Object

~~|
~~*---IdMapBase

**Directly known subclasses:**

- Bootstrap, Corr, IdMap, IdMapCounts, IdMapDiff, IdMapDiffCounts, JointIdMap, JointUniquePairs, UniquePairs

public static class `IdMapBase`

extends `Object`

IdMapBase is an abstract object encapsulating a data frame with at least two columns, the first one (primary) containing character string s identifying the ID under consideration (unprot accessions ID or acc, Entrez Gene ID etc) and the rest of columns containing the various information associated with a given primary ID for a particular DB service.
IdMapBase(DF=NULL, name="", primaryKey=NULL, secondaryKey=NULL, ...)

Arguments
DF A data.frame consisting of two columns (primary and secondary IDs) from which the IdMap object is to be created.
name A character string representing the name of the given IdMap object. Default is ""
primaryKey The primary identifier type from which the ID conversion is performed. If NULL (default) then the input data frame first column name is used and if it is not available defaults to 'From'.
secondaryKey The secondary identifier type to which conversion is performed. Default is NULL.
...

Fields and Methods
Methods:

aligned Checks if two IdMapBase objects match on column names and primary ID set.
as.data.frame Retrieves a data frame encapsulated within the given IdMapBase object.
dim Retrieves dimensions of data frame encapsulated within the given IdMapBase object.
dimnames Retrieve or set the dimnames of data frame encapsulated within the given IdMapBase object.
getName Get the name a given IdMapBase object.
primaryIDs Retrieves the primary IDs for a given IdMapBase object.
primaryKey Retrieves a primary key for a given IdMapBase object.
secondaryKey Retrieves a secondary key for a given IdMapBase object.

Methods inherited from Object:
$, $<-, [], [[, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, 
equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Alex Lisovich, Roger Day

Examples
DF<-array(0,dim=c(5,2));
obj<-IdMapBase(DF,primaryKey="primary",secondaryKey="secondary");

IdMapCounts The IdMapCounts class
IdMapCounts

Description

Package: IdMappingAnalysis
Class IdMapCounts

Object
~~|  
~~+--IdMapBase
~~~~~~|  
~~~~~~~+--IdMapCounts

Directly known subclasses:

public static class IdMapCounts  
extends IdMapBase

An IdMapCounts object enapsulates a data.frame where the first column contains the primary ID set while the rest of columns contain the counts of secondary IDs for each Id Map in a given idMapList object, one column per ID Map, each ID Map related column having a name representing the given DB data source (i.e. 'NetAffx', 'EnVision' etc.) The constructor creates the IdMapCounts object from the list of ID Maps aligned by the primary IDs and primary and secondary keys. The easiest way to obtain the list of properly aligned IdMap objects is to create a JointIdMap object from a set of un-aligned ID maps and then invoke the getIdMapList() method on this object. The IdMapCounts object can also be created directly from JointIdMap object by using the JointIdMap.$getCounts() method.

Usage

IdMapCounts(idMapList=NULL, verbose=FALSE, ...)

Arguments

idMapList The list of ID Maps aligned on primary IDs.
verbose If TRUE enables diagnostic messages. Default is FALSE
...
Not used.

Fields and Methods

Methods:

getStats Retrieves a set of unique counts of secondary IDs.
plot Compute and plot the (inversed) ecdf for each ID Map count entry within the IdMapCounts object.

Methods inherited from IdMapBase:
[. aligned, as.data.frame, dim, dimnames, getName, primaryIDs, primaryKey, secondaryKey

Methods inherited from Object:
$, $<-, [., [<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstan-
IdMapDiff

The IdMapDiff class

Description

Package: IdMappingAnalysis

Class IdMapDiff

Object

~~|
~~+--IdMapBase
~~~|
~~~~~~+--IdMapDiff

Directly known subclasses:

public static class IdMapDiff
extends IdMapBase

IdMapDiff constructor implements most time consuming step in comparing two DBs and the structure itself stores the results in a compact form. The IdMapDiff object encapsulates a data.frame the first column of which contains the primary IDs and the rest of columns contain a disjoint representation of the ID Map pair in the form of 3 columns <A-A*B,A*B,B-A*B>, where A and B are secondary ID lists for ID Maps A and B. This class is separated from the IdMapDiffCounts in anticipation of being used by various processing pipelines in a future.
**IdMapDiff**

**Usage**

```
IdMapDiff(idMap1=NULL, idMap2=NULL, pairNames=c("First", "Second"), verbose=FALSE, ...)```

**Arguments**

- **idMap1**
  The first ID Map object on which IdMapDiff object is constructed.
- **idMap2**
  The second ID Map object on which IdMapDiff object is constructed.
- **pairNames**
  The character vector of length 2 representing the names of the ID Map pair. Default is c('First', 'Second').
- **verbose**
  If TRUE enables diagnostic messages.
- **...**
  Not used.

**Fields and Methods**

- **Methods:**
  No methods defined.

- **Methods inherited from IdMapBase:**
  [, aligned, as.data.frame, dim, dimnames, getName, primaryIDs, primaryKey, secondaryKey

- **Methods inherited from Object:**
  $, $<-, [[, [/[, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInvalidationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

**Author(s)**

Alex Lisovich, Roger Day

**Examples**

```r
# get primary IDs from an msms experiment set
IDs <- IdMapBase$primaryIDs(examples$msmsExperimentSet);

# create JointIdMap object aligned by primaryIDs
jointIdMap <- JointIdMap(examples$identDfList, primaryIDs=IDs);

# get IdMap list aligned of two ID maps aligned by primaryIDs
idMaps <- jointIdMap$getIdMapList(verbose=TRUE);

# create IdMapDiff object
diffs <- IdMapDiff(idMaps["NetAffx_F"], idMaps["DAVID_Q"];
diffs[1:10];

# create IdMapDiff object directly from JointIdMap
diffs <- jointIdMap$getDiff("NetAffx_F", "DAVID_Q", verbose=TRUE);
```
The `IdMapDiffCounts` class handles statistics on `IdMapDiff` object. `IdMapDiffCounts` object encapsulates a data frame with row names corresponding to the primary IDs and 6 columns subdivided into pairs `<match(TRUE/FALSE),count>` each pair corresponding to the disjoint events `<A-A*B,A*B,B-A*B>`, where A and B are secondary ID lists for ID Maps A and B from the `IdMapDiff` object. The `pairNames` attribute of the `IdMapDiffCounts` contains the names of the source ID Map pair from which the `IdMapDiff` object was created.

Usage

```r
IdMapDiffCounts(idMapDiff=NULL, verbose=FALSE, ...)
```

Arguments

- `idMapDiff` The `IdMapDiff` on which `IdMapDiffCounts` is created. Default is `NULL`.
- `verbose` If `TRUE` enables diagnostic messages. Default is `FALSE`.
- `...` Not used.

Fields and Methods

Methods:

- `getCompoundEvents` Get compound events.
- `getCompoundGroups` Get counts for each compound event in `IdMapDiffCounts`.
- `plot` Produce a fountain plot representing the quantitative relationship of the compound events.
- `summary` Get a compound event counts summary report.

Methods inherited from `IdMapBase`:
JointIdMap

The Joint ID Map class

Description

Package: IdMappingAnalysis

Class JointIdMap

Object

|--IdMapBase

--------|

--------+---JointIdMap

Directly known subclasses:

public static class JointIdMap extends IdMapBase

JointIdMap is an object encapsulating a data.frame containing the primary ID set in a first column while the rest of columns containing the sets of secondary IDs, each column corresponding to a particular Id Map, keeping all Id Maps properly aligned.
JointIdMap

Usage

JointIdMap(idMapList=list(), primaryIDs=NULL, name="", verbose=FALSE, ...)

Arguments

idMapList  The list of ID Maps on which the JointData is constructed.
primaryIDs  The optional character vector of primary IDs on which an additional intersection and reordering are performed.
name  The optional name of a given JointIdMap object. Default is ”
verbose  if TRUE enables diagnostic messages. Default is @FASLE.
...  Not used

Fields and Methods

Methods:

as.data.frame  Retrieve a data frame encapsulated within the given JointIdMap object.
diffCounts.plot  Interactive wrapper for IdMapDiffCounts$plot.
edf.plot  Interactive wrapper for IdMapCounts$plot.
getCounts  Create an IdMapCounts object.
getDiff  Create an IdMapDiff object.
getIdMapList  Create an Id Map list from a JointIdMap object.
getMapNames  Get the names of IdMap objects encapsulated within the given JointIdMap object.
getMatchInfo  Get match table(s) for a given set of primary IDs.
getUnionIdMap  Create a union IdMap.

Methods inherited from IdMapBase:
[, aligned, as.data.frame, dim, dimnames, getName, primaryIDs, primaryKey, secondaryKey

Methods inherited from Object:
$, $<-, [., [<-., as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstanceationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Roger Day, Alex Lisovich

Examples

jointIdMap<-JointIdMap(examples$identDfList);

jointIdMap$primaryKey();
jointIdMap$secondaryKey();

jointIdMap[1:10,];
JointUniquePairs

**The JointUniquePairs class**

**Description**

Package: IdMappingAnalysis

**Class JointUniquePairs**

Object

```
~~|--
~~~|---IdMapBase
~~~~~~|---JointUniquePairs
```

**Directly known subclasses:**

public static class JointUniquePairs extends IdMapBase

UniquePairsMatch object encapsulates a data frame the first two columns of which contain the unique pairs corresponding to the merge (union) of all ID Maps in consideration while the rest of columns contains the match (logical value) between the merged unique pairs set and a unique pairs set specific to the particular ID Map (‘d8’, ‘enV’, ‘netAffx’ etc), one column per Id Map. Used in combination with correlation related data objects (CorrData, Corr, Mixture etc.) to aid in on-fly processing related to some classification by a particular match group. The UniquePairsMatch constructor creates an object from the UniquePairs and an ID Map list computing the match (inclusions) for each particular ID Map.

**Usage**

`JointUniquePairs(uniquePairs=NULL, idMapList=NULL, name='', verbose=FALSE, ...)`

**Arguments**

- **uniquePairs**: UniquePairs object on which a UniquePairsMatch is created or a `data.frame` complying with the UniquePairs class internal data frame format. In case the UniquePairs object is used as a first argument, it’s typically obtained from the JointIdMap object by invoking JointIdMap$getUnionIdMap()
- **idMapList**: the list of ID Maps on which the match is performed during the UniquePairs-Match object creation The idMapList typically obtained through the call to the JointIdMap.getIdMapList() of the same JointIdMap object as for the first argument to ensure that both arguments are properly aligned.
- **name**: A character string representing the name of the given IdMap object. Default is “”
- **verbose**: If TRUE enables diagnostic messages. Default is FALSE.
- **...**: Not used.
Fields and Methods

Methods:

- `boxplot`: Draw a basic boxplot based on a given JointUniquePairs object and external data.
- `corr.boxplot`: Boxplot of correlations by match group.
- `corr.plot`: Plot the density distributions for a set of correlation objects.
- `do(glm)`: Compute linear regression for the given set of ID Maps.
- `getBootstrap`: Create Bootstrap object from JointUniquePairs object and two experiment sets.
- `getCorr`: Extract a set of correlation objects from given JointUniquePairs object.
- `getCorrData`: Create CorrData object from the JointUniquePairs object and two experiment sets.
- `getCorrDataFrame`: Merge JointUniquePairs and Corr objects into a single data frame.
- `getMatchInfo`: Get the names of UniquePairs objects encapsulated within the given JointUniquePairs object.
- `getMixture`: Extract mixture model object from JointUniquePairs and Corr objects.
- `getMapNames`: Extract the unity UniquePairs object from a given JointUniquePairs object.
- `getUniquePairs`: Interactive boxplot of correlations by match group.
- `interactive.corr.boxplot`: Interactive boxplot of correlation densities.
- `interactive.corr.plot`: Interactive plot of correlation densities.
- `interactive.mixture.boxplot`: Interactive boxplot of mixture component probabilities by match group.
- `interactive.mixture.plot`: Interactive plot of mixture model components.
- `mixture.boxplot`: General purpose JointUniquePairs interactive plot function.
- `mixture.plot`: Plot the correlation densities of the empirical fit, mixture fit and each.
- `subsetCorr`: Subset the Corr object.
- `subsetData`: Subset data on a UniquePairsMatch object.
- `subsetGroups`: Get a JointUniquePairs subset.
- `subsetGroups`: Get a JointUniquePairs subset.

Methods inherited from IdMapBase:
- `[`, `as.data.frame`, `dim`, `dimnames`, `getName`, `primaryIDs`, `primaryKey`, `secondaryKey`

Methods inherited from Object:
- `$`, `as.character`, `attach`, `attachLocally`, `clearCache`, `clearLookupCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `gc`, `getEnvironment`, `getFieldModifier`, `getFieldModifiers`, `getFields`, `getInstanceTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `objectSize`, `print`, `registerFinalizer`, `save`

Author(s)

Alex Lisovich, Roger Day

Examples

```r
# create JointIdMap
jointIdMap <- JointIdMap(examples$identDfList);

# create unique pairs from the union of all IdMaps within JointIdMap
pairs <- as.UniquePairs(jointIdMap$getIdMapList(), verbose=TRUE);

# create JointUniquePairs object
jointPairs <- JointUniquePairs(pairs, jointIdMap$getIdMapList(), verbose=TRUE);
jointPairs[1:10,]
```
The Misc class

Description

Package: IdMappingAnalysis
Class Misc

Object

~~|--
~~+--Misc

Directly known subclasses:

public static class Misc
extends Object

Serves as a wrapper for various miscallaneous functions used throughout the package defined as static methods of the Misc class.

Usage

Misc()

Fields and Methods

Methods:

CsvList.merge  Pairwise merge of two string vectors.
interleave     Interleave two matrices by columns.
to.base        Convert number to a numeric vector of a given base.
to.binary.logical  Convert number to a vector of logicals.
to.index.expr  Convert expression into index expression for a given list or data frame object.
words          Convert space delimited string to a vector of words.

Methods inherited from Object:
$, $<-, [], [[]<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach,
equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstance-
tiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Alex Lisovich, Roger Day
**Description**

Package: IdMappingAnalysis

Class Mixture

Object

```
~~|
```

```
```

```
```

**Directly known subclasses:**

```
public static class Mixture
extends Object
```

The constructor creates a model from a single Corr object using the number of clusters defined by G determining the optimal number of clusters by default and optionally using the Fisher transform.

**Usage**

```
Mixture(corr=NULL, G=c(1:5), Fisher=FALSE, verbose=FALSE, ...)
```

**Arguments**

```
corr  Corr object on wich mixture modeling is performed.
G     number of components in mixture model. If G is a vector, the optimal number of components is determined. G is a vector (1:5) by default.
Fisher if TRUE, the Fisher transform of correlation data is performed before the model is fitted. Default is FALSE.
verbose if TRUE enables diagnostic messages. Default is FALSE.
...    Not used.
```

**Value**

The resulting Mixture object encapsulates a data member `.model` containing the results of mixture modeling represented by the list with following components:

```
corr      the correlation data
clust     the clustering results data structure returned by Mclust()
sd        standard deviation derived from clust$parameters$variance$sigmasq
density   the correlation density distribution
marginalDensity  the marginal density
```
Mixture

Fields and Methods

Methods:
subset
clust Retrieve the clustering results data structure.
getData Extract mixture component data from the Mixture object.
getStats Get mixture component model summary info.
plot Plot the results of mixture modeling.
primaryKey Retrieves a primary key for a given Mixture object.
secondaryKey Retrieves a secondary key for a given Mixture object.

Methods inherited from Object:
S, $<-$, $[.[], [[<-$, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)
Alex Lisovich, Roger Day

Examples
mixture<-Mixture(examples$corr,G=c(1:4),Fisher=TRUE,verbose=TRUE);
class(mixture);
names(mixture$.model)

Subset The Subset class

Description
Package: IdMappingAnalysis
Class Subset

Object
~~|~~
~~|--|---Subset

Directly known subclasses:

public static class Subset
extends Object

Serves as a wrapper for data frame subsetting functions defined as static methods of the Subset class.

Usage
Subset(...)
UniquePairs

Arguments

... Not used.

Fields and Methods

Methods:

- **byColNames**: Extract subset of columns from a data frame or a list of data frames.
- **byColumn**: Extract subset of rows from a data frame or a list of data frames.
- **byRow**: Extract subset of columns from a data frame or a list of data frames.
- **byRowNames**: Extract subset of columns from a data frame or a list of data frames.

Methods inherited from Object:

$ S$. $S<-$. $[[], [[<-$. as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Alex Lisovich, Roger Day

---

UniquePairs The UniquePairs class

Description

Package: IdMappingAnalysis

Class UniquePairs

Object

```
---
```

```
```

Directly known subclasses:

```

class UniquePairs extends IdMapBase
```

The alternative representation of an IdMap suitable for performing the correlation related processing. Contains a data frame with two columns, each row of which represents a unique pair <primary ID, secondary ID> where primary ID corresponds to the primaryIDs of an ID Map and secondary ID corresponds to a single ID from a list of comma separated secondary IDs within the corresponding ID Map. The column names correspond to the primary/secondary keys of an Id Map (‘acc’ and ‘probeset’ for example)
Usage

UniquePairs(DF=NULL, name="", primaryKey=colnames(DF)[1], secondaryKey=colnames(DF)[2], ...)

Arguments

DF A data.frame consisting of two columns (primary and secondary IDs) from which the UniquePairs object is to be created.
name A character string representing the name of the given UniquePairs object. Default is "
primaryKey The name of the primary (first) column of a data.frame encapsulated within the UniquePairs object. If missing then the input data frame first column name is used and if it is not available defaults to 'From'.
secondaryKey The name of secondary (second) column in an ID Map. If missing then the input data frame second column name is used and if it is not available defaults to 'To'.
... Not used.

Fields and Methods

Methods:

as.IdMap Convert the UniquePairs object into the IdMap object.
asCreate a UniquePairs object from a single IdMap or a list of IdMap objects.
equalsc Check if two unique pairs data structures are identical.
getMatches Get the logical vector of pair matches of the given UniquePairs object.
swapKeys Swap the primary and secondary key columns.
unique Extract unique elements.

Methods inherited from IdMapBase:
[, aligned, as.data.frame, dim, dimnames, getName, primaryIDs, primaryKey, secondaryKey

Methods inherited from Object:
$, $<-, [], [[]<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstanceationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

Author(s)

Alex Lisovich, Roger Day

Examples

DF<-matrix(
c("P25685","200664_s_at",
"P25685","200666_s_at",
"Q6ZV71","205208_at",
"Q6ZV71","215798_at",
"P05164" , "203948_s_at"
),ncol=2,nrow=5,byrow=TRUE);
colnames(DF)<-c("Uniprot","Affy");
uniquePairs<-UniquePairs(DF);
Index

*Topic classes
  Bootstrap, 2
  Corr, 3
  CorrData, 5
  DataFilter, 6
  Display, 7
  IdMap, 12
  IdMapBase, 13
  IdMapCounts, 14
  IdMapDiff, 16
  IdMapDiffCounts, 18
  JointIdMap, 19
  JointUniquePairs, 21
  Misc, 23
  Mixture, 24
  Subset, 26
  UniquePairs, 27
*Topic data
  examples, 8
  aligned, 14
  as.data.frame, 14, 20
  as.IdMap, 28
  as.list, 13
  as.MultiSet, 6
  as.UniquePairs, 13
  Bootstrap, 2, 13
  boxplot, 22
  byColNames, 27
  byColumn, 27
  byRow, 27
  byRowNames, 27
  character, 12, 14, 20, 21, 28
  clust, 26
  copy, 8
  Corr, 3, 13
  corr.boxplot, 22
  corr.plot, 22
  CorrData, 5
  create, 8, 28
  CsvList.merge, 23
  data.frame, 3, 4, 12, 14–16, 19, 21, 28
  DataFilter, 6
  diffCounts.plot, 20
  dim, 14
  dimnames, 14
  Display, 7
  do.apply, 7
  do.glm, 22
  ecdf.plot, 20
  equals, 28
  examples, 8
  expectedUtility, 9
  FALSE, 2, 4, 5, 15, 18, 21, 24
  fisherTransform, 7
  fisherTransformInverse, 7
  fisherTransformJacobean, 7
  fit2clusters, 10
  getBootstrap, 22
  getCompoundEvents, 18
  getCompoundGroups, 18
  getCorr, 22
  getCorrData, 22
  getCorrDataFrame, 22
  getCounts, 13, 20
  getData, 4, 26
  getDiff, 20
  getExperimentSet, 6
  getIdMapList, 20
  getMapNames, 20, 22
  getMatch, 28
  getMatchInfo, 20, 22
  getMixture, 22
  getName, 14
  getSampleNames, 6
  getStats, 15, 26
  getUnionIdMap, 20
  getUniquePairs, 4, 6, 22
  IdMap, 12, 13
  IdMapBase, 2–4, 12, 13, 15, 16, 18, 19, 21, 27
  IdMapCounts, 13, 14
  IdMapDiff, 13, 16

30
INDEX

IdMapDiffCounts, 13, 18
interactive.corr.boxplot, 22
interactive.corr.plot, 22
interactive.mixture.boxplot, 22
interactive.mixture.plot, 22
interactive.plot, 6, 22
interleave, 23
JointIdMap, 13, 19
JointUniquePairs, 13, 21
line.loess, 8
line.unsorted, 8
list, 15, 20, 24
logTen, 7
merge, 13
minAvgCountConstraint, 7
minCountConstraint, 7
minCountGroupConstraint, 7
Misc, 23
Mixture, 24
mixture.boxplot, 22
mixture.plot, 22
NULL, 14, 18
Object, 2, 3, 5–7, 12, 13, 15, 16, 18, 19, 21,
23, 24, 26, 27
plot, 3, 4, 6, 15, 18, 26
primaryIDs, 14
primaryKey, 6, 14, 26
progressMsg, 8
removeNASeries, 7
secondaryKey, 6, 14, 26
Subset, 26
subsetCorr, 22
subsetData, 22
subsetGroups, 22
summary, 18
swapKeys, 13, 28
textBoundingBox, 8
to.base, 23
to.binary.logical, 23
to.index.expr, 23
TRUE, 2, 4, 5, 15, 17, 18, 20, 21, 24
unique, 28
UniquePairs, 13, 27
words, 23
zoom.pars, 8