The *IdMappingAnalysis* package in Bioconductor: Critically comparing identifier maps retrieved from bioinformatics annotation resources

Alex Lisovich †, Roger S. Day †‡

November 7, 2012

†Department of Biomedical Informatics, †‡ Department of Biostatistics, University of Pittsburgh

1 Introduction

With increasing frequency, studies of biological samples include processing on two (or more) high-throughput platforms. Each platform produces a large set of features, each labeled by an identifier. It is one thing to merge the data by sample, simply combining the features on both platforms into a single data set. However, exploiting the full biological significance of the data depends on linking a feature from one platform with a feature on the other, where the pair of features are believed to be closely causally related to one another biologically. Examples include an ID from an mRNA expression array and an ID from a proteome-wide mass spectrometry data set, when the mRNA species is believed to produce the corresponding protein when translated. Many bioinformatics resources facilitate this mapping between two types of ID. But they tend to disagree with each other, sometimes dramatically.

The *IdMappingAnalysis* package helps an analyst to critically compare the identifier maps retrieved from various annotation resources, to help choose the best ID mapping resource to use in merging the data. One can also use this technique to choose the best ID filtering strategy for removing dubious ID's.

As presented here, this methodology focuses on comparing several ID mapping resources, but it is actually quite general. One can also use it to optimize the selection of a threshold for some quality index for mappings or for individual features, or to compare other data preparation and preprocessing methods.

This document begins with an overview of the setting and capabilities, followed by detailed examples and explanations, and finishing with a few notes about the package architecture.

2 Overview of capabilities

2.1 The setting and the problem statement

Suppose we have two data frames containing high-throughput data on the same set of samples. The two data frames are from two different platforms, designed to quantitate different but related types of molecular features. The two sets of feature names are identifiers, or IDs, which may be platform-specific, such as a probe set ID on an expression array, or may correspond to a standard identifier set, such as gene name, UniProt ID, or microRNA identifier or accession.

In light of widespread annotation errors in bioinformatics resources, our problem is to evaluate different data processing options, either for mapping between the two ID lists (ID mapping, IDM) or for selecting IDs more likely to be correctly annotated (ID filtering, IDF). Our strategy is to use these two data frames merged across both sample and ID, using different IDM or IDF methods, and compare the methods by comparing the distributions of correlations between mapped features. (Correlation is just one possible quality indicator; others may be more suitable for some applications.)

The *IdMappingAnalysis* package characterizes individual ID maps, creates data-free comparisons of two ID maps, and creates comparative evaluations of two ID maps in light of merged experimental data for a set of samples. In detail, the three capabilities provided are:

1. **Characterizing ID maps.** Each ID map is described by the empirical distribution of the number of secondary IDs. The ID map may be obtained by sending a list of primary IDs from an experimental data set to a bioinformatics

resource, to retrieve for each primary ID the list of secondary IDs that according to this resource, correspond to the primary ID.

Section 3.4 demonstrates this capability.

2. **Comparing two ID maps (without data).** The next step in understanding the behavior of different mapping resources is to compare them. Two given mapping resources A and B probed with the same list of primary IDs will each provide a list of secondary IDs. This data is easily converted into a set of ID pairs for each resource. Each primary-secondary ID pair is classified as appearing in "A only", "B only", or both. Then the counts in these categories are summarized across all ID pairs appearing in either A or B.

   Section 3.5 demonstrates this capability.

3. **Evaluating and comparing two ID maps based on integrating experimental results.** To obtain an evaluation of the relative quality of two mapping resources A and B, we require two data sets reporting results from two different high-throughput platforms 1 and 2 applied to the same samples. The names of the features of platform 1 are primary IDs; the names of the features of platform 2 are secondary IDs. For each ID pair reported by either A or B, we imagine that the corresponding pair of features falls into one of three categories:

   (a) both features are correctly identified and mapped between platforms, and truly biologically coupled (for example, correlated) as expected,

   (b) both features are correctly identified and mapped, but biologically decoupled or dysregulated in the sample group under study, due to causes unaccounted-for up to now,

   (c) one or both features are incorrectly identified, or incorrectly mapped to each other.

   Comparing two strategies A and B for ID mapping, the best of the two should be enriched for the first and second categories of feature pairs, relative to the third. With enough samples and feature pairs, the identity of the best strategy should become clear. The main technique used is mixture modeling followed by regression modeling.

   Once a preferring strategy for ID mapping is selected, the same strategy also serves to comparatively evaluate two different criteria or algorithms for filtering features on one of the platforms.

   Section 3.6 demonstrates this capability.

3 Examples in detail

The examples in this section recapitulate the creation of tables and figures in an article [4] comparing ID mapping resources for semantically merging mass spectrometry shotgun proteomics data with expression microarray data. The results are not identical, primarily because the data analyzed here constitute a subset of the full data set, for purposes of storage and speed.

3.1 Package setup

We begin loading the package.

```
> library(IdMappingAnalysis);
```

This also loads *rChoiceDialogs*, needed for some interactive features.

3.2 Collecting the ID mapping data

The starting point of the analysis is a vector of bioinformatics identifiers, the primary IDs. This list may be the list of features for a particular high-througut platform, such as the probe sets on a specific expression array. Alternatively, the primary IDs may come from a specific experiment, for example a mass spectrometry "shotgun" experiment, where protein UniProt accession identifiers are delivered by a specific algorithm matching a peptide spectrum to a peptide and thence to a protein.

With a list of primary IDs in hand, the task is to map these to a set of secondary IDs, and the problem statement is to select the best bioinformatics mapping resource to use for this task. For each primary ID, each resource is queried for secondary ID matches. The results from each service are assembled into an `IdMap` object. The `IdMap` objects for the services are then assembled into a single `JointIdMap` object.

In the following example, the mapping services of interest are accessible using the *IdMappingRetrieval* Bioconductor package [5]. Alternatively, one can use the ID map constructor for any mapping services not handled by *IdMappingRetrieval*. (OSX users should install *IdMappingRetrieval* via the command
`biocLite("IdMappingRetrieval", type="source")`
because of Java version issues.)

Depending on the mapping service, the process of retrieving a mapping may take significant time and disk space. To make this overview document practical to assemble in packaging and to use as a tutorial, the subsequent code snippets use subsets of the maps pre-acquired using the code listed above and placed into the *IdMappingAnalysis* package data section.

```
> options(width = 110)

> # Initialize the ID mapping retrieval system
> library(IdMappingRetrieval);
> Annotation$init();
> AnnotationAffx$setCredentials(user=MY_AFFY_USERNAME,
+       password=MY_AFFY_PASSWORD,verbose=FALSE);
> # Create a service manage object encapsulating default ID retrieval services.
> svm <- ServiceManager(ServiceManager$getDefaultServices());
> # Retrieve the ID Map list for selected array type and services.
> identDfList <- getIdMapList(svm,arrayType="HG-U133_Plus_2",
+       selection=names(svm$getServices()),verbose=TRUE);
> class(identDfList)
> class(identDfList)[[1]]
```

In this example, however, we use the data provided with the package.

Notice that `identDfList` is an ordinary list, but each member of identDfList is an object of class `IdMap`. The first column of an `IdMap` data object is the primary ID. There is also a secondary ID column containing the retrieved matches for each primary ID value, pasted into a single string with `collapse=","` as the separator.

3.3 Preparing a `JointIdMap` object from multiple ID maps

```
> # A list of ID maps to be analysed.
> names(examples$identDfList)

[1] "NetAffx_F"  "NetAffx_Q"  "DAVID_Q"    "DAVID_F"    "EnSembl_F"  "EnVision_Q"

> head(examples$identDfList[[1]], 5)

       Uniprot                                          Affy
Q96RW7  Q96RW7                                     235944_at
Q8WUU4  Q8WUU4                                     238607_at
Q86VP3  Q86VP3    1555823_at,1555824_a_at,212778_at,34406_at
Q93075  Q93075                          1565580_s_at,203648_at
Q16539  Q16539 202530_at,210449_x_at,211087_x_at,211561_x_at

> # Define the primary and secondary IDs to work with.
> primaryIDs <- IdMapBase$primaryIDs(examples$msmsExperimentSet)
> head(primaryIDs)   ### UniProt IDs for proteins.

[1] "P04264" "P13645" "P35908" "P08729" "P08727" "P25705"
```

```
> secondaryIDs <- IdMapBase$primaryIDs(examples$mrnaExperimentSet);
> head(secondaryIDs)  ### Affymetrix probeset IDs.

[1] "208930_s_at" "231060_at"   "212937_s_at" "200656_s_at" "229743_at"   "200968_s_at"

> # Construct a JointIdMap object which combines
> # the various ID maps, aligned by the union of the primary ID vectors.
> jointIdMap <- JointIdMap(examples$identDfList,primaryIDs,verbose=FALSE);
> names(getMethods.Class(JointIdMap)[["JointIdMap"]])  ### Methods specific to JointIDMap

[1] "as.data.frame"   "diffCounts.plot" "ecdf.plot"       "getCounts"       "getDiff"
[6] "getIdMapList"    "getMapNames"     "getMatchInfo"    "getUnionIdMap"

> str(jointIdMap$as.data.frame())  ##  Structure of the data field of JointIdMap

'data.frame':        2395 obs. of  7 variables:
 $ Uniprot   : chr  "P04264" "P13645" "P35908" "P08729" ...
 $ NetAffx_F : chr  "205900_at" "207023_x_at,210633_x_at,213287_s_at" "207908_at" "209016_s_at,214031_s_at"
 $ NetAffx_Q : chr  "205900_at" "207023_x_at,213287_s_at,210633_x_at" "207908_at" "209016_s_at,214031_s_at"
 $ DAVID_Q   : chr  "205900_at" "213287_s_at,210633_x_at,207023_x_at" "207908_at" "1558394_s_at,209016_s_at
 $ DAVID_F   : chr  "205900_at" "210633_x_at,213287_s_at,207023_x_at" "207908_at" "1558393_at,209016_s_at,1
 $ EnSembl_F : chr  "205900_at" "213287_s_at,210633_x_at,207023_x_at" "207908_at" "209016_s_at" ...
 $ EnVision_Q: chr  "205900_at" "207023_x_at,210633_x_at,213287_s_at" "207908_at" "209016_s_at" ...
 - attr(*, "name")= chr ""
 - attr(*, "secondaryKey")= chr "Affy"
```

Note that the first column of the `JointIdMap` data object is the primary ID. The other columns are the retrieved matches (pasted with ",", as with IdMap), for each of the ID mapping resources.

We can also reverse the mapping, as follows.

```
> identDfListReversed <- lapply(examples$identDfList, function(identDf)
+     IdMap$swapKeys(IdMap(identDf)))
> jointIdMapReversed <- JointIdMap(identDfListReversed, secondaryIDs, verbose=FALSE)
```

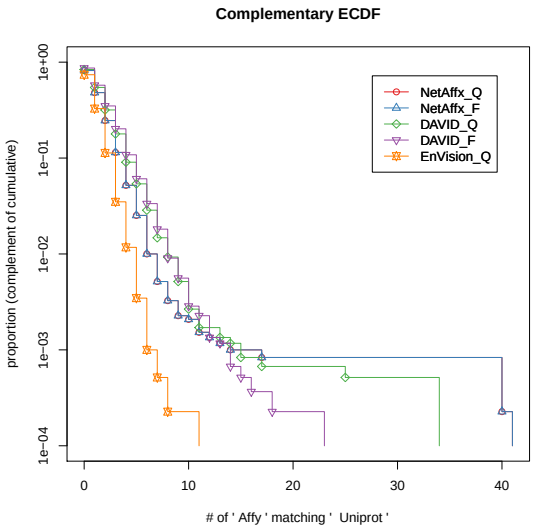
(Reversing the mapping is not necessarily its own inverse; an ID with no matches will disappear in the reverse of the reverse.)

3.4 Individual characterization of ID maps.

The following code creates the analogues of Table 1 and Figure 1 in [4].

```
> # Assemble secondary ID counts object for a given set of DB's
> mapCounts <- getCounts(jointIdMap,
+           idMapNames=c("NetAffx_Q","NetAffx_F","DAVID_Q","DAVID_F","EnVision_Q"),
+           verbose=FALSE);
> #  This shows the structure of the contents of an IdMapCounts object.
> str(mapCounts$as.data.frame())

'data.frame':        2395 obs. of  6 variables:
 $ Uniprot   : Factor w/ 2395 levels "A0A580","A0AUH1",..: 266 363 515 319 318 438 284 931 864 372 ...
 $ NetAffx_Q : int  1 3 1 2 2 2 1 2 1 4 ...
 $ NetAffx_F : int  1 3 1 2 2 2 1 2 1 4 ...
 $ DAVID_Q   : int  1 3 1 4 1 2 1 2 1 5 ...
 $ DAVID_F   : int  1 3 1 4 2 3 1 2 2 5 ...
 $ EnVision_Q: int  1 3 1 1 1 1 1 2 1 3 ...
 - attr(*, "name")= chr ""
```

```
> names(getMethods.Class(mapCounts)[["IdMapCounts"]])

[1] "getStats" "plot"

> #  Tabulating the number of returned secondary IDs per primary ID, aligned across services.
> statsByMatchCount <- mapCounts$getStats(summary=FALSE,verbose=FALSE);
> statsByMatchCount[1:6,1:10];

  cnts NetAffx_Q NetAffx_Q (%) NetAffx_F NetAffx_F (%) DAVID_Q DAVID_Q (%) DAVID_F DAVID_F (%) EnVision_Q
0    0       374          15.6       374          15.6     332        13.9     269        11.2        539
1    1       730          30.5       730          30.5     629        26.3     626        26.1        923
2    2       559          23.3       559          23.3     526        22.0     515        21.5        554
3    3       349          14.6       349          14.6     350        14.6     366        15.3        239
4    4       187           7.8       187           7.8     244        10.2     254        10.6         85
5    5        90           3.8        90           3.8     112         4.7     141         5.9         36

> #  ...and the same results with a simplified summary.
> mapCounts$getStats(summary=TRUE,cutoff=3,verbose=FALSE);

             0    1    2    3   >3 Max.count Total.count
NetAffx_Q 15.6 30.5 23.3 14.6 16.0        41        5005
NetAffx_F 15.6 30.5 23.3 14.6 16.0        41        5005
DAVID_Q   13.9 26.3 22.0 14.6 23.2        34        5855
DAVID_F   11.2 26.1 21.5 15.3 25.9        23        6167
EnVision_Q 22.5 38.5 23.1 10.0  5.9        11        3394

> #  A plot of empirical cdf's of the secondary ID counts
> par(mfrow = c(1, 2));
> mapCounts$plot();  ### Or alternatice syntax:  plot(mapCounts)
```



3.5 Comparison of two ID maps

Now we compare two ID maps for each primary ID, then display the results across primary IDs with tabular and graphical summaries. The following code creates the analogues of Figures 2-5 in [4].

```
> diffsBetweenMap <- jointIdMap$getDiff("DAVID_Q","EnVision_Q",verbose=FALSE);
> class(diffsBetweenMap)
```

```
[1] "IdMapDiff" "IdMapBase" "Object"
```

```
> diffCounts <- IdMapDiffCounts(diffsBetweenMap,verbose=FALSE);
> names(getMethods(IdMapDiffCounts)[["IdMapDiffCounts"]])
```

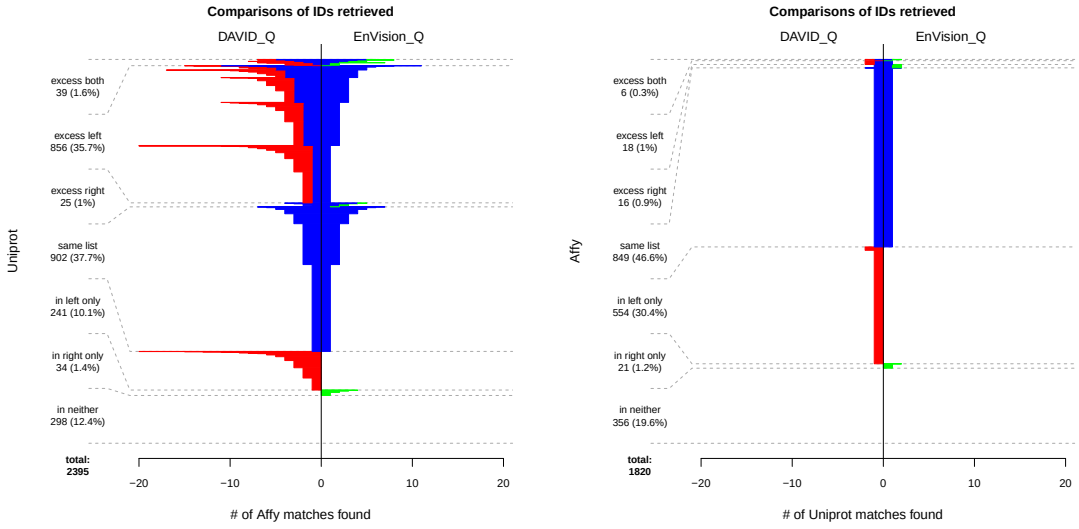
```
[1] "getCompoundEvents" "getCompoundGroups" "plot"              "summary"
```

```
> ###  We characterize the differences betwen the two ID maps.
> diffCounts$summary(verbose);
```

```
              counts
excess.both       39
excess.left      856
excess.right      25
same.list        902
in.left.only     241
in.right.only     34
in.neither       298
```

```
> par(mfrow = c(1, 2));
> diffCounts$plot(adj=0.1,sides=1);
> # The same result is achieved more succinctly (for a different pair of ID mapping resources in this examp
> # summary(jointIdMap$diffCounts.plot(c("DAVID_Q", "EnVision_Q"),
> #                                    adj=0.1,sides=1,verbose=FALSE));
> # We can also characterize the reverse  mapping created above.
> summary(jointIdMapReversed$diffCounts.plot(c("DAVID_Q", "EnVision_Q"),
+                                    adj=0.1,sides=1,verbose=FALSE));
```

```
              counts
excess.both        6
excess.left       18
excess.right      16
same.list        849
in.left.only     554
in.right.only     21
in.neither       356
```



Interactive use is available through a wrapper menu, invoked by the arg "loop".

```
> jointIdMap$diffCounts.plot("loop",adj=0.1,sides=1,verbose=FALSE);
```

3.6 Performing a comparative evaluation of two ID maps based on integrating experimental results from two platforms.

In this section, we utilize data from the two experiments to create and compare evaluations of several ID mapping resources, as manifest in the semantically integrated data. By "semantic integration" we mean the process of selecting, for each ID pair obtained from any of the ID maps, the two corresponding features in the two experiments, and merging them by sample. It is semantic in the sense that the ID maps are intended to represent true biological relations.

3.6.1 Preparing a data set for each ID pair

In this example, the data-based comparison of ID maps uses a mass spectrometry experiment and an Affymetrix microarray experiment on the same samples. In the interest of reducing package storage, a subset of features is selected. In the interest of anonymity, data are jittered. The primary IDs are the protein UniProt IDs from the mass spectrometry experiment. The MS/MS data is first filtered. We keep columns for which the average of counts across all samples is greater than 0.5. We keep rows for which all the data are not missing. The secondary IDs

```
> msmsExperimentSet <- DataFilter$do.apply(examples$msmsExperimentSet,
+      byRows=TRUE,filterFun=DataFilter$minAvgCountConstraint,filtParams=0.52,verbose=FALSE);
> dim(examples$msmsExperimentSet)

[1] 2395   99

> dim(msmsExperimentSet)

[1] 2395   99

> msmsExperimentSet <- DataFilter$removeNASeries(msmsExperimentSet,byRows=TRUE,verbose=FALSE);
> dim(msmsExperimentSet)

[1] 469  99

> mrnaExperimentSet <- examples$mrnaExperimentSet
> # We log10-transform the mRNA signal data.
> mrnaExperimentSet[,-1] <- log10(mrnaExperimentSet[,-1]);
> #  The primary IDs are now defined by the experiment.
> primaryIDs_from_dataset <- IdMapBase$primaryIDs(msmsExperimentSet);
> secondaryIDs <- IdMapBase$primaryIDs(mrnaExperimentSet);
> #  The method name primaryIDs() is unfortunate! This will be changed.
>
> # Create a JointIdMap object as before,
> # but with a  primary ID vector reduced by the filtering step above.
> jointIdMap_from_dataset <- JointIdMap(examples$identDfList, primaryIDs_from_dataset, verbose=FALSE);
```

3.6.2 Creating a merged data set for each ID pair

Next we create an object of class `UniquePairs` containing a data frame of all ID pairs from all the ID maps being analyzed. For each ID pair, we want to calculate the correlation of the corresponding features from the two experimental data sets. Therefore we prepare a `CorrData` object which collates the data to facilitate rapidly calculating all the correlations in one step.

```
> uniquePairs <- as.UniquePairs(
+     getUnionIdMap(jointIdMap_from_dataset,verbose=FALSE),secondaryIDs);
> str(uniquePairs$as.data.frame())

'data.frame':        1541 obs. of  2 variables:
 $ Uniprot: chr  "P04264" "P13645" "P13645" "P13645" ...
 $ Affy   : chr  "205900_at" "207023_x_at" "210633_x_at" "213287_s_at" ...
 - attr(*, "name")= chr "NetAffx_F"
```

```
> corrData <- CorrData(uniquePairs,
+     examples$msmsExperimentSet,examples$mrnaExperimentSet,verbose=FALSE);
> str(corrData)

Classes 'CorrData', 'Object'  atomic [1:1] NA
  ..- attr(*, ".env")=<environment: 0x12754530>

> names(getMethods(CorrData)[["CorrData"]])

[1] "as.MultiSet"      "getExperimentSet" "getSampleNames"   "getUniquePairs"   "interactive.plot"
[6] "pack.experiments" "plot"             "primaryKey"       "secondaryKey"

> # (The method as.MultiSet() is deprecated.)
```

Next, we create an object of class *JointUniquePairs* that records, for each unique pair, which of the ID maps reported it. This will be used later for regression analyses to determine which ID maps best predict good correlations. It is also handy for subselecting secondary ID features.

```
> # create pairs match object from unique pairs and joint ID map object
> jointUniquePairs <- JointUniquePairs(uniquePairs,
+     getIdMapList(jointIdMap_from_dataset,verbose=FALSE),verbose=FALSE);
> str(jointUniquePairs$as.data.frame())

'data.frame':        1541 obs. of  8 variables:
 $ Uniprot   : chr  "P04264" "P13645" "P13645" "P13645" ...
 $ Affy      : chr  "205900_at" "207023_x_at" "210633_x_at" "213287_s_at" ...
 $ NetAffx_F : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
 $ NetAffx_Q : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
 $ DAVID_Q   : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
 $ DAVID_F   : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
 $ EnSembl_F : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
 $ EnVision_Q: logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
 - attr(*, "name")= chr ""
```

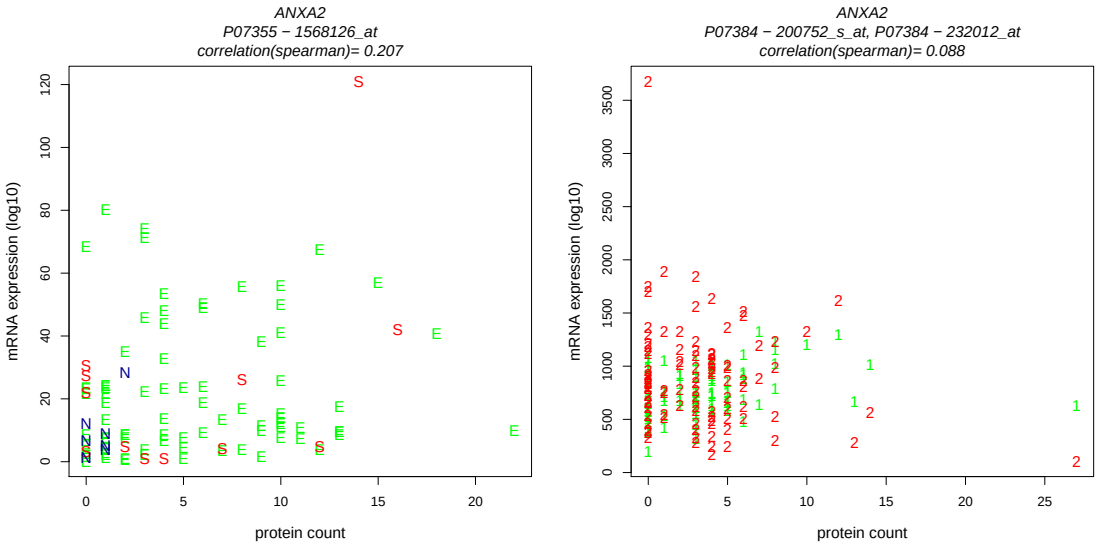
3.6.3 Scatterplots for data corresponding to a selected ID pair

For a particular primary ID, in this example P07355, we want to view correlations and scatterplots for this feature together with the features on the other experiment corresponding to mapped secondary IDs: in this example, the probeset IDs which are reported in one or more ID pairs to correspond to P07355. The following code creates the analogues of Figures 6 and 7 in [4].

```
> # cross-correlation matrix of the  expression signals for probesets mapped with UniprotAcc="P07355"
>
> idMatchInfo <- jointIdMap_from_dataset$
+                       getMatchInfo(IDs="P07355",
+                                idMapNames=c("NetAffx_Q", "DAVID_Q", "EnVision_Q"))[[1]]
> print(idMatchInfo)

           201590_x_at 210427_x_at 213503_x_at 1568126_at 210876_at 211241_at
NetAffx_Q  201590_x_at 210427_x_at 213503_x_at 1568126_at
DAVID_Q    201590_x_at 210427_x_at 213503_x_at 1568126_at 210876_at 211241_at
EnVision_Q 201590_x_at 210427_x_at 213503_x_at

> data_Uniprot = corrData$getExperimentSet(modality="Uniprot",
+       IDs="P07355")
> data_Affy <- corrData$getExperimentSet(modality="Affy",
+       IDs=colnames(idMatchInfo));
> data <- cbind(t(data_Uniprot[,-1]), t(data_Affy[,-1]))
> cor(data,method="spearman");
```

```
              P07355 201590_x_at 210427_x_at 213503_x_at  1568126_at   210876_at  211241_at
P07355      1.0000000  0.53381294   0.5027493   0.5332040  0.20676100  0.31792059 0.27333674
201590_x_at 0.5338129  1.00000000   0.9780171   0.9804143  0.07808794  0.14324697 0.42917210
210427_x_at 0.5027493  0.97801707   1.0000000   0.9758239  0.10087607  0.11997603 0.40942693
213503_x_at 0.5332040  0.98041428   0.9758239   1.0000000  0.10774312  0.16306870 0.42916572
1568126_at  0.2067610  0.07808794   0.1008761   0.1077431  1.00000000 -0.07625211 0.04957742
210876_at   0.3179206  0.14324697   0.1199760   0.1630687 -0.07625211  1.00000000 0.19408024
211241_at   0.2733367  0.42917210   0.4094269   0.4291657  0.04957742  0.19408024 1.00000000
```

```
> # Scatterplot  for Uniprot="P07355" (annexin 2), probe set ID="1568126_at".
> # Patient outcomes guide symbols and colors.
> par(mfrow = c(1, 2));
> corrData$plot(input=list(c("P07355","1568126_at")),
+          outcomePairs=examples$outcomeMap,proteinNames="ANXA2",
+      cex=1.2,cex.main=1.2,cex.lab=1.2,cols=c("green","red","darkblue"));
> # scatterplot with outcome for Uniprot="P07384" (annexin 2)
> # for all matching probesets without outcome
> corrData$plot(input="P07384",proteinNames="ANXA2",
+      cex=1.2,cex.main=1.2,cex.lab=1.2,cols=c("green","red","darkblue"));
```



3.7 Exploring merged data interactively using the high level wrappers

There are also interactive menu-driven versions.

```
> # interactive scatterplot with a single primary ID (uniprot) and outcomes
> corrData$interactive.plot(c("P07355"),
+       outcomePairs=examples$outcomeMap,proteinNames="ANXA2");
> # interactive scatterplot with a single primary ID (uniprot) and without outcomes
>  corrData$interactive.plot(c("P07355"),proteinNames="ANXA2");
> # interactive scatterplot with multiple probeset IDs (uniprot) and without outcomes
> corrData$interactive.plot(c("P07355","P07384","P09382"));
> # interactive scatterplot with all available probeset
> #IDs (uniprot) and without outcomes
> corrData$interactive.plot();
```

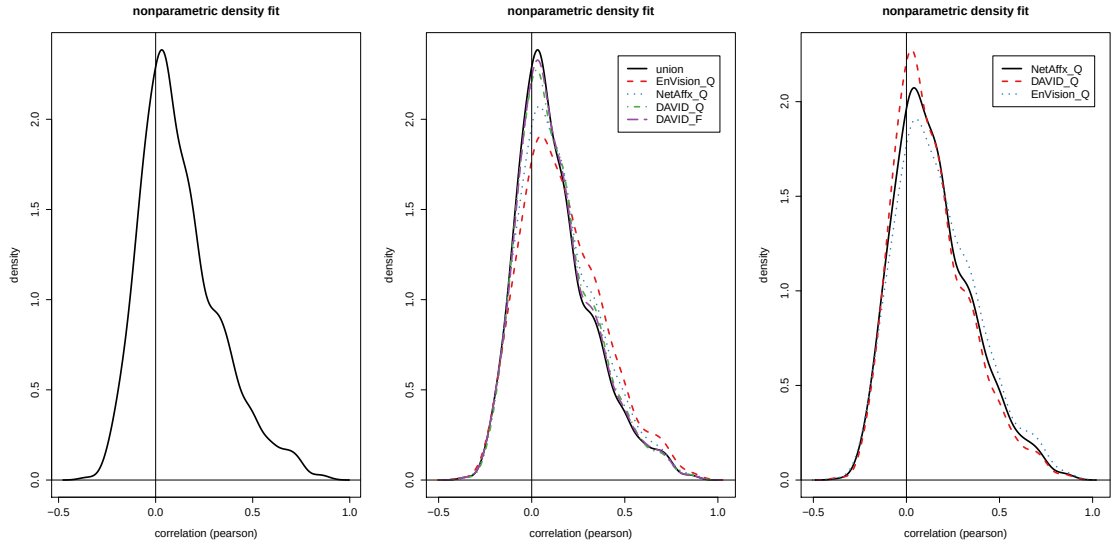
3.8 Evaluating and comparing Id Map quality using correlations

We can calculate correlations for all the ID pairs, and use them as "model quality" measures, for the model in which protein abundance should be proportional to transcript abundance. There are many things to criticize about this model, but by looking across a large set of ID pairs, the correlations can provide insight as to which of the ID mapping methods is working the best.

3.8.1 Calculating and displaying the correlations

The following code creates the analogues of Figures 8-10 in [4]. The density fit for correlations is for the union of all ID maps from all the ID mapping resources under study. Each unique ID pair counts just once regardless of the number of ID maps that contain it. The left-most graph shows a fairly tendency towards positive correlations. Its appearance is consistent with a mixture of a substantial proportion of "noise" correlations with a substantial proportion of "real" correlations. The next two graphs show more detail. The middle graph overlays individual correlation densities for each ID mapping resource. It shows all the curves, while the right-hand graph overlays individually chosen correlation densities for each ID mapping resource.

```
> par(mfrow = c(1, 3));
> corr <- Corr(corrData,method="pearson",verbose=FALSE);
> corr[1:6, ]

  Uniprot        Affy     pearson
1  P04264   205900_at -0.03646315
2  P13645 207023_x_at -0.05323715
3  P13645 210633_x_at -0.09455382
4  P13645 213287_s_at -0.05994931
5  P35908   207908_at -0.10343738
6  P08729 209016_s_at  0.70042064

> ## Left-hand plot: density estimate for all correlations.
> corr$plot(cex=1.2,cex.main=1.4,cex.lab=1.2,cex.legend=1.2);
> ## Center plot: individual density estimates for selected Id mapping resources.
> corrSet <- getCorr(jointUniquePairs,corr,
+       groups=c("union","EnVision_Q","NetAffx_Q","DAVID_Q","DAVID_F"),
+       full.group=TRUE,verbose=FALSE);
> names(corrSet)

[1] "union"      "EnVision_Q" "NetAffx_Q"  "DAVID_Q"    "DAVID_F"

> Corr$plot(corrSet,cex=1.2,cex.main=1.4,cex.lab=1.2,cex.legend=1.2);
> ## Right-hand plot: individual density estimates for selected Id mapping resources.
> corrSet <- jointUniquePairs$corr.plot(corr,
+       idMapNames=c("NetAffx_Q","DAVID_Q","EnVision_Q"),
+       plot.Union=FALSE, subsetting=TRUE, verbose=FALSE,
+       cex=1.2, cex.main=1.4 ,cex.lab=1.2, cex.legend=1.2);
> names(corrSet)

[1] "NetAffx_Q"  "DAVID_Q"    "EnVision_Q"
```



To construct a correlation density plot interactively, the following method call provides an alternative.

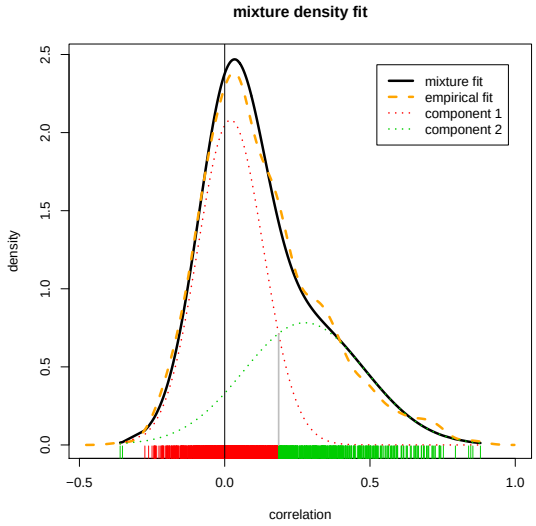
```
> jointUniquePairs$interactive.corr.plot(corr,verbose=FALSE);
```

3.8.2 Alternative endpoint: posterior probability of a "good" correlation base on a mixture model

The density smooths suggest a mixture model, with a component cntered at correlation=zero and another located in the positive range of correlations. We fit a mixture model using the package **mclust**. This generates a posterior probability for each component (assuming a 50-50 prior for component membership). The posterior probability of the positive component is arguably a better measure than the correlation itself for purposes of identifying factors that predict a correct ID mapping. Only large really correlations matter, and the difference between a negative correlation and a zero correlation is nil. While the transformation from correlation to the posterior probability is approximately monotone (as currently calculated), it stretches the important part of the correlation scale and compresses the unimportant part.

There are shortcomings to be addressed. Currently the model is fit without assuming equal variances; consequently, if the fitted variances are much different, then the transformation to posterior probability is not monotone. So far no problems appear to have arisen in practice because of this. It is easily addressed by requiring equal variances for the two mixture components, but this is not yet implemented in this package. Second, by rights the zero-centered component should be centered exactly at zero. The option to fix one of the means is not readily available through *mclust*. Again, so far in practice the left-most component has always had its mean very close to zero, making it a plausible representation for correlations due to noise.

```
> # create and plot the mixture model for number of components = 2
> mixture <- Mixture(corr,G=2,verbose=FALSE);
> par(mfrow=c(1, 2));
> mixture$plot();
> mixture$getStats();

                comp.1     comp.2
mean        0.02037288  0.2754195
sd          0.11280934  0.2099948
weight      0.58815209  0.4118479
range.low  -0.27475372 -0.3598423
range.high  0.18532543  0.8801130
```



There are a few options. One can vary the number of components in the mixture, and one can restrict to a subset of the ID mappers (the `groups` parameter).

```
> # Create and plot the mixture model determining
> #the optimal number of components)
> # for a given DB subset treating the subset as a full group
> mixture.subset <- jointUniquePairs$getMixture(corr, groups=c("NetAffx_Q","DAVID_Q","EnVision_Q"),
+       full.group=TRUE,G=c(1:5),verbose);
> mixture.subset <- jointUniquePairs$mixture.plot(corr,
+       idMapNames=c("NetAffx_Q","DAVID_Q","EnVision_Q"),
+       subsetting=TRUE,G=c(1:5),verbose=FALSE);
```

The mean of component #1 is close to zero. The standard deviations of the components are somewhat different. In principle, then, a correlation near -1 could be classified as component #2; but so far this does not happen in practice.

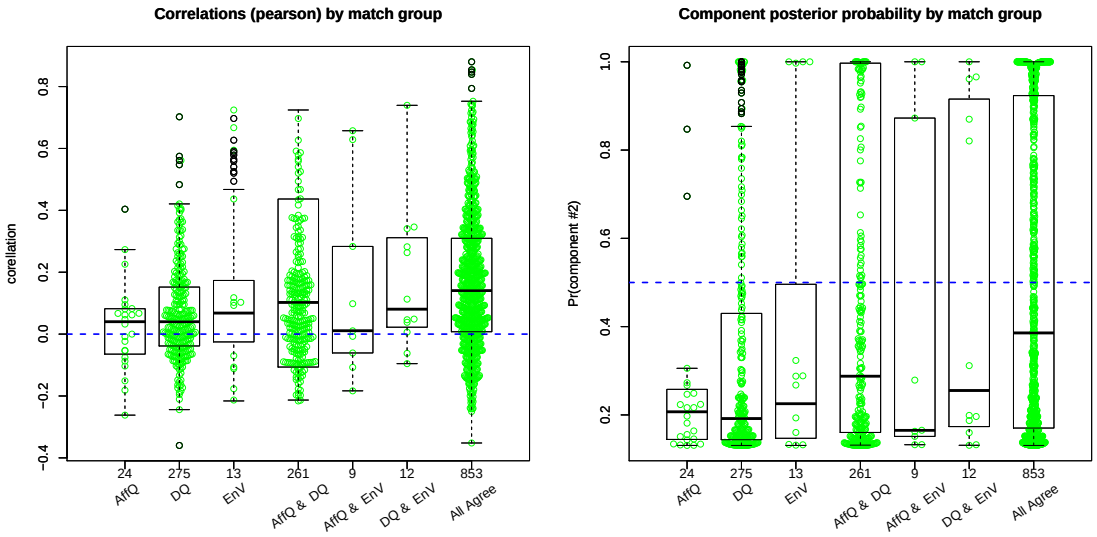
```
> # retrieve the mixture parameters
> mixture$getStats();
```

```
                comp.1     comp.2
mean        0.02037288  0.2754195
sd          0.11280934  0.2099948
weight      0.58815209  0.4118479
range.low  -0.27475372 -0.3598423
range.high  0.18532543  0.8801130
```

We can compare the correlation distributions of the subsets of ID pairs defined by which of the mapping services reports the pair. The analogous boxplot figure using the compenent #2 posterior probability emphasizes the differences for large positive correlations, and is probably more informative. Note how the individual groups are accessible through list item names with a Boolean-like syntax.

```
> par(mfrow = c(1, 2));
> # Choose the mapping services, and define the corresponding short names to use in the figure.
> mappingServicesToPlot <- list("NetAffx_Q"="AffQ","DAVID_Q"="DQ","EnVision_Q"="EnV");
> # Plot correlation probability distributions by match group
> boxplotResult_correlation = jointUniquePairs$corr.boxplot(
+   corr, idMapNames=mappingServicesToPlot, subsetting=TRUE,
+   srt=35, col.points="green");
> # The following extracts the correlations for the group of ID pairs
```

```
> # reported by DQ and EnV but not by AffQ.
> boxplotResult_correlation$response.grouped$`!AffQ & DQ & EnV`

 [1] -0.062215756  0.340786098  0.346203727 -0.095519260  0.282205104  0.006357974  0.038169869  0.7394379
 [9]  0.112938352  0.046416742  0.263356579  0.048522628

> # Plot posterior second component probability distributions by match group
> boxplotResult_mixtureProb = jointUniquePairs$mixture.boxplot(
+   corr, idMapNames=mappingServicesToPlot, subsetting=TRUE,
+   plot.G=2, srt=35, col.points="green");
> # invisible by default.
> boxplotResult_mixtureProb$response.grouped$`!AffQ & DQ & EnV`

 [1] 0.1324723 0.9611375 0.9657365 0.1312042 0.8699137 0.1598080 0.1872751 1.0000000 0.3116469 0.1964275
[11] 0.8206111 0.1989159
```



Again, there is an interactive menu-driven version.

```
> # plot the results of mixture fit interactively choosing the DB subset
>  interactive.mixture.plot(jointUniquePairs,corr,verbose=FALSE);
```

3.8.3 Comparisons of mapping services

We can use regression modeling to investigate many different possible predictors of correlation, where correlation is a model quality measure, thought of as a reflection of the chance that the ID pair is correct. Here, for our quality measure for the linear model, instead of the correlation between the features, we use the posterior probability of belonging to the right-hand positive-correlation mixture component.

```
> # Perform regression analysis relating which mapping services predict good correlations.
>
> fit<-jointUniquePairs$do.glm(corr$getData(),
+        idMapNames=c("DAVID_Q","EnVision_Q","NetAffx_Q"));
> coefficients(summary(fit));

                 Estimate Std. Error  t value     Pr(>|t|)
(Intercept)    0.03590490 0.01698235 2.114248 3.465479e-02
DAVID_QTRUE    0.03399767 0.01890522 1.798322 7.232214e-02
EnVision_QTRUE 0.08022383 0.01286295 6.236812 5.759386e-10
NetAffx_QTRUE  0.01668979 0.01510508 1.104913 2.693703e-01
```

```
> # Perform regression analysis using the second mixture component as the outcome variable.
> qualityMeasure <- mixture$getData(G=2) ## 2nd component posterior probability
> fitLinear <- jointUniquePairs$do.glm(qualityMeasure,
+       idMapNames=c("DAVID_Q","EnVision_Q","NetAffx_Q"));
> coefficients(summary(fitLinear));

                 Estimate Std. Error  t value     Pr(>|t|)
(Intercept)    0.24200027 0.02745614 8.814068 3.197375e-18
DAVID_QTRUE    0.07076364 0.03056492 2.315191 2.073361e-02
EnVision_QTRUE 0.14179941 0.02079612 6.818551 1.317365e-11
NetAffx_QTRUE  0.03138424 0.02442106 1.285130 1.989405e-01

> #To do this directly:
> fitLinear <- with(as.data.frame(jointUniquePairs),
+       glm(qualityMeasure ~ DAVID_Q + EnVision_Q + NetAffx_Q))
> coefficients(summary(fitLinear));

                 Estimate Std. Error  t value     Pr(>|t|)
(Intercept)    0.24200027 0.02745614 8.814068 3.197375e-18
DAVID_QTRUE    0.07076364 0.03056492 2.315191 2.073361e-02
EnVision_QTRUE 0.14179941 0.02079612 6.818551 1.317365e-11
NetAffx_QTRUE  0.03138424 0.02442106 1.285130 1.989405e-01

> # Another variation:
> fitHitCount <- with(as.data.frame(jointUniquePairs),
+       glm(qualityMeasure ~ I(DAVID_Q + EnVision_Q + NetAffx_Q)))
> coefficients(summary(fitHitCount));

                                    Estimate  Std. Error  t value     Pr(>|t|)
(Intercept)                       0.21856034 0.020411477 10.70772 7.426553e-26
I(DAVID_Q + EnVision_Q + NetAffx_Q) 0.08647295 0.008391697 10.30458 4.002377e-24

> fitHitCount$dev - fitLinear$dev

[1] 0.8650637
```

With the caveat that this is a reduced illustrative data set compared to the full data reported in [4], the linear model seems to indicate that EnVision is the best predictor, DAVID less so, and NetAffx the least, although comparison with "hit count" model counteracts that impression. Using the 2nd component probability seems somewhat more sensitive than using the correlations as the endpoint.

The regression analyses above are unweighted analyses. However, the error variance is not constant. We can supply weights reflecting the variances as estimated either by the Fisher's formula for the variance of a correlation coefficient (ignoring the non-normality), or by a bootstrap. The latter are provided by the `Bootstrap` class within the *IdMapping-Analysis* package.

```
> bootstrap<-Bootstrap(corrData,Fisher=TRUE,verbose=FALSE);
> str(as.data.frame(bootstrap))

'data.frame':        1541 obs. of  5 variables:
 $ Uniprot: chr  "P04264" "P13645" "P13645" "P13645" ...
 $ Affy   : chr  "205900_at" "207023_x_at" "210633_x_at" "213287_s_at" ...
 $ corr   : num  -0.0365 -0.0532 -0.0946 -0.0599 -0.1034 ...
 $ sd     : num  0.0815 0.0663 0.0595 0.0792 0.0732 ...
 $ bias   : num  -0.00757 0.00236 0.00216 0.0059 -0.00351 ...
 - attr(*, "name")= chr ""
```

```
> # Scatterplot of the estimated standard deviation versus the observed correlation.
> bootstrap$plot(new.plot=TRUE,file.copy=TRUE,copy.zoom=2,bg="white");
> # One can use these estimates as weights.
> fitLinear <- with(as.data.frame(jointUniquePairs),
+       glm(qualityMeasure ~ DAVID_Q + EnVision_Q + NetAffx_Q,
+          weights=(as.data.frame(bootstrap)$sd) ^ (-2))
+ )
> coefficients(summary(fitLinear));

                Estimate Std. Error  t value     Pr(>|t|)
(Intercept)    0.23660124 0.02593410 9.123172 2.222497e-19
DAVID_QTRUE    0.03786448 0.02897795 1.306666 1.915218e-01
EnVision_QTRUE 0.16178227 0.02003639 8.074424 1.355787e-15
NetAffx_QTRUE  0.02342066 0.02334937 1.003053 3.159930e-01

> # Another variation:
> fitHitCount <- with(as.data.frame(jointUniquePairs),
+       glm(qualityMeasure ~ I(DAVID_Q + EnVision_Q + NetAffx_Q),
+          weights=(as.data.frame(bootstrap)$sd) ^ (-2))
+ )
> coefficients(summary(fitHitCount));

                                   Estimate  Std. Error   t value     Pr(>|t|)
(Intercept)                       0.1908368 0.019518402  9.777278 6.043314e-22
I(DAVID_Q + EnVision_Q + NetAffx_Q) 0.0855038 0.008127019 10.520930 4.787227e-25

> fitHitCount$dev - fitLinear$dev

[1] 208.198
```

With this weighting, the evidence in favor of EnVision's predictive power is much stronger.

To compare two services directly, the following code is illustrative. One can perform two-sample tests on the subgroups of ID pairs that exclusively are reported by one but not the other service. Performing a Wilcoxon test, it will not usually matter whether the endpoint is the correlation or Pr(component #2). But a t test should be more powerful in detecting a difference where it matters: large correlations.

```
> affyOnly = boxplotResult_mixtureProb$response.grouped$`AffQ & !DQ & !EnV`
> envOnly = boxplotResult_mixtureProb$response.grouped$`!AffQ & !DQ & EnV`
> davidOnly = boxplotResult_mixtureProb$response.grouped$`!AffQ & DQ & !EnV`
> allThree = boxplotResult_mixtureProb$response.grouped$`AffQ & DQ & EnV`
> t.test(affyOnly, envOnly)

        Welch Two Sample t-test

data:  affyOnly and envOnly
t = -1.5645, df = 16.846, p-value = 0.1363
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.42716623  0.06354892
sample estimates:
mean of x mean of y
0.2731738 0.4549825

> wilcox.test(affyOnly, envOnly)
```

```
        Wilcoxon rank sum test

data:  affyOnly and envOnly
W = 108, p-value = 0.1321
alternative hypothesis: true location shift is not equal to 0
```

```
> t.test(affyOnly, davidOnly)
```

```
        Welch Two Sample t-test

data:  affyOnly and davidOnly
t = -1.1514, df = 28.912, p-value = 0.259
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.15954857  0.04462228
sample estimates:
mean of x mean of y
0.2731738 0.3306369
```

```
> wilcox.test(affyOnly, envOnly)
```

```
        Wilcoxon rank sum test

data:  affyOnly and envOnly
W = 108, p-value = 0.1321
alternative hypothesis: true location shift is not equal to 0
```

```
> t.test(affyOnly, allThree)
```

```
        Welch Two Sample t-test

data:  affyOnly and allThree
t = -4.7468, df = 26.009, p-value = 6.553e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3305250 -0.1307701
sample estimates:
mean of x mean of y
0.2731738 0.5038214
```

```
> wilcox.test(affyOnly, allThree)
```

```
        Wilcoxon rank sum test with continuity correction

data:  affyOnly and allThree
W = 6442, p-value = 0.001938
alternative hypothesis: true location shift is not equal to 0
```

The last two tests show the increased power from using the 2nd component probability instead of the correlation as the quality measure.

```
> head(jointUniquePairs$as.data.frame() )
```

	Uniprot	Affy	NetAffx_F	NetAffx_Q	DAVID_Q	DAVID_F	EnSembl_F	EnVision_Q
1	P04264	205900_at	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
2	P13645	207023_x_at	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
3	P13645	210633_x_at	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
4	P13645	213287_s_at	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
5	P35908	207908_at	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
6	P08729	209016_s_at	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

```
> pairs <- jointUniquePairs$as.data.frame()
> pairsNotReportedByNetAffx = pairs[
+   (!pairs$NetAffx_Q & (pairs$DAVID_Q | pairs$EnVision_Q))
+   , c("Uniprot", "Affy")]
> head(pairsNotReportedByNetAffx)

   Uniprot         Affy
8   P08729 1558394_s_at
9   P08729   1558393_at
24  P14625    239451_at
27  P05787    230429_at
28  P05787    229879_at
51  P62805    234960_at
```

4 High-level wrapper methods and interactive data exploration

The `JointIdMap`, `JointUniquePairs` and `CorrData` classes provide a set of high level wrapper methods supporting interactive menu-driven selection of arguments and performance of multiple processing steps within a single method invocation. The interactive session can generate either single or multiple data sets/plots corresponding to the user input selections or these methods can be used non-interactively. The user is protected from some of the complexity of multiple processing steps. The following functionality is supported through the methods arguments or by selection within the GUI:

- multiple or single GUI-based data exploration sessions
- optional subsetting on ID mapping identifiers or ID mapping resources
- plotting into existing or automatically created new device device
- copying a plot into the file with a specified "zoom" (magnification factor).

5 The package architecture

A few notes on the architecture of the package may be helpful. The package *R.oo* provides a reasonably simple class/method framework based on the S3 class system.

5.1 Classes

The `IdMapBase` class is fundamental to the *IdMappingRetrieval* package. It contains a data frame whose columns include two columns designated as primaryKey and secondaryKey, whose values are two ID types under study. An `IdMapBase` object is usually obtained from methods of the package *IdMappingRetrieval*. Classes which extend `IdMapBase` are:
`IdMap, IdMapCounts, JointIdMap, IdMapDiff, IdMapDiffCounts, UniquePairs, JointUniquePairs, Corr, Bootstrap`

They all inherit `as.data.frame()`.

An `IdMap` object has a `primaryKey` which is truly a key: values are unique. All classes use the same `as.data.frame` method, except `JointIdMap`, for which the notion of `secondaryKey` needs modification, because there are multiple `secondaryKey` columns.

Viewing the names of methods specific to each of these classes, use the *R.oo* function `getMethods.Class` and this syntax:

`names(getMethods.Class(JointIdMap)[["JointIdMap"]]).`

To see the content of any method, use the usual syntax for S3 class methods, for example `diffCounts.plot.JointIdMap`. To see the names of fields in a class (including "hidden" fields), use syntax like this:

`getFields.Class(JointIdMap, private=TRUE).`

To access the value of a field in an instance, use syntax like this: `jointIdMap$.secondaryKey`.

The rest of the classes are just containers for static auxiliary functions:

- **Subset:** handles data subsetting and merging operations

- **DataFilter:** provides various algorithms for data preprocessing
- **Display:** provides utility functions for graphics support, including rescaling of plots for export to files.
- **Misc:** contains miscellaneous helper functions.

Contents of the classes

6 Session information

```
> toLatex(sessionInfo())
```

- R version 2.15.2 (2012-10-26), `i386-w64-mingw32`
- Locale: `LC_COLLATE=C`, `LC_CTYPE=English_United States.1252`, `LC_MONETARY=English_United States.1252`, `LC_NUMERIC=C`, `LC_TIME=English_United States.1252`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: IdMappingAnalysis 1.2.1, R.methodsS3 1.4.2, R.oo 1.10.1, rChoiceDialogs 1.0.4, rJava 0.9-3
- Loaded via a namespace (and not attached): RColorBrewer 1.0-5, boot 1.3-7, mclust 4.0, tools 2.15.2

7 References

[**1**] Liu G., NetAffx: Affymetrix probesets and annotations. Nucleic Acids Res. 2003, 31(1):82-6.

[**2**] Huang D.W., Sherman B.T., Lempicki R.A. (2008) Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. Nat. Protoc., doi: 10.1038/nprot.2008.211.

[**3**] Reisinger F., Corpas M., Hancock J., Hermjakob H., Birney E., Kahlem P.. ENFIN - An Integrative Structure for Systems Biology. Data Integration in the Life Sciences Lecture Notes in Computer Science, 2008, Volume 5109/2008, 132-143, DOI: 10.1007/978-3-540-69828-9_13

[**4**] Day R.S., McDade K.K., Chandran U., Lisovich A., Conrads T.P., Hood B., Kolli V.S.K., Kirchner D., Litzi T., Maxwell G.L.. Identifier mapping performance for integrating transcriptomics and proteomics experimental results. BMC Bioinformatics, 2011. 12: p. 213.

[**5**] Day R.S., Lisovich A.. DAVIDQuery: Retrieval from the DAVID bioinformatics data resource into R. R package version 1.12.0., in Bioconductor Release, 2.9. 2010.

[**6**] Lisovich A., Day R.S., ENVISIONQuery: Retrieval from the ENVISION bioinformatics data portal into R. R package version 1.2.0, in Bioconductor Release 2.9, 2011

[**7**] Lisovich A., Day R.S., The IdMappingRetrieval package in Bioconductor: Collecting and caching identifier mappings from online sources. R package Version 1.1.0, in Bioconductor Development, 2.10. 2011.

[**8**] Lisovich A., Day R.S., The rChoiceDialogs package in CRAN: Collection of portable choice dialog widgets. R package version 1.0.1 Published 2011-01-15.

[**9**] Bengtsson H., The R.oo package - Object-oriented programming with references using standard R code, Talk at the DSC 2003 conference, Wienna, March 22, 2003.