Annotating data with affycoretools and biomaRt

James W. MacDonald

February 7, 2007

1 Introduction

Most of the functions in *affycoretools* have been designed to annotate Affymetrix probesets using the annotation packages produced by the Biocore data team, in concert with the *annaffy* package. This paradigm works quite well for those Affymetrix chips that are popular enough to have an annotation package, but if there isn't a package for a given chip, one is left to either try to build a package using *AnnBuilder* (which can be a daunting task), or to try to annotate their probesets using *biomaRt*.

The *biomaRt* package is getting to be much more user friendly, but there still exists a gap between getting the annotation for a set of probesets and producing a finished product that can be presented to someone (e.g., an HTML table). Steffen Durinck (the maintainer of *biomaRt*) was kind enough to add functionality to his package that will allow one to easily take the output from his package straight into the `htmlpage` function of the *annotate* package to create HTML tables. I have since written some functions that are analogous to the existing *affycoretools* functions, but that use *biomaRt* and `htmlpage`.

For any first time users of *affycoretools*, I would direct you to the 'Using affycoretools' vignette first, which is designed to be an introductory text for this package.

2 Getting Started

First, a note about the *biomaRt* package. The functions in this package are designed to interface with an online BioMart database, usually one hosted at `http://www.biomart.org`. There are two interfaces; the default interface uses the *RCurl* package to connect to the database, but there is another interface that uses the *RMySQL* package to connect. The default interface is nice for doing interactive annotation of a small number of probesets, but can get *exceedingly* slow for anything more than say, 30 or 40 probesets. Therefore, I strongly recommend using the *RMySQL* interface if at all possible. Note that the kind folks in Seattle usually have a compiled version of *RMySQL* that can be installed using `biocLite`.

Given that most *affycoretools* functions are designed to help make an analysis more efficient, the most reasonable way to show how things work is to emulate an analysis. For this, we will be using the data in R_Home/library/affycoretools/examples. We can assume that these data are four different samples (which we will call A – D), that were run in triplicate (biological replicates). These data come from the Affymetrix HG-focus array, which has a BioC annotation package. However, instead of using the normal probeset mapping provided by Affy, we can use a re-mapped cdf where the mappings are based on Entrez Gene. This re-mapped cdf doesn't have an annotation package, so we will have to use *biomaRt* to annotate.

First, we compute expression values.

```
> library(affycoretools)
> eset <- justRMA(cdfname = "hsfocushsentrezg7cdf")
```

Now we model the data using *limma*. Using a cell means model (without an intercept, denoted by the `~0` in the call to `model.matrix`) is necessary for most of the functions in the *affycoretools* package, as the names for the output are in general extracted from the contrasts matrix.

```
> library(limma)
> design <- model.matrix(~0 + factor(rep(1:4, each = 3)))
> colnames(design) <- LETTERS[1:4]
> contrast <- makeContrasts(A - B, C - D, levels = design)
> fit <- lmFit(eset, design)
> fit2 <- contrasts.fit(fit, contrast)
> fit2 <- eBayes(fit2)
```

Here we are using A-B and C-D as names for the contrasts matrix. I usually give more descriptive names that will help somebody not involved in the analysis figure out what is being compared. After fitting the model and computing contrasts, we can output all significant probesets with an adjusted p-value less than 0.05 using `limma2biomaRt`.

```
> limma2biomaRt(eset, fit2, design, contrast, species = "hsapiens",
+     pfilt = 0.05, interactive = FALSE)
```

There are many more arguments for this function, but the defaults should be good for most uses (I hope). This will create HTML tables for the two comparisons we made.

Note that if there is no replication and one is just selecting probesets based on fold change, the `foldFiltBM` function can be used. In addition, if one has a vector of probe IDs to annotate, the `probes2tableBM` function will output HTML tables as well.

One might ask what probesets are differentially expressed in common between the two comparisons (or one might be interested in those probesets that are not in common). We can visualize this using a Venn diagram.

```
> dt <- decideTests(fit2, method = "nestedF")
> rslt <- vennCounts2(dt)
> vennDiagram(rslt)
```

Figure 1 shows the Venn diagram. Now if we want to output tables containing the probesets in each cell of the Venn diagram, we can use `vennSelectBM`.

```
> vennSelectBM(eset, design, rslt, contrast, fit2, species = "hsapiens")
```

Note that there are two helper functions being used by these functions; `linksBM`, and `annBM`. These functions have two purposes; first, if called with no arguments, they list the type of annotation that is possible to use to create hyperlinks (`linksBM`), or that can be used for annotating without hyperlinking (`annBM`). For `linksBM`, this list includes all the annotation sources that `htmlpage` is able to hyperlink. This list can always be increased, and I am open to suggestions for other databases that one might want to link to. For `annBM`, the list is comprised of things that I normally use, and could also be extended to other annotation data.

The second purpose for these two functions is to list which of the *possible* annotation sources actually exist at a particular BioMart. This is done by passing a `mart` object to either function. This purpose is necessary because not all BioMart databases contain the same information. For instance, some of the BioMart databases don't have gene symbols.

```
> annBM()
```

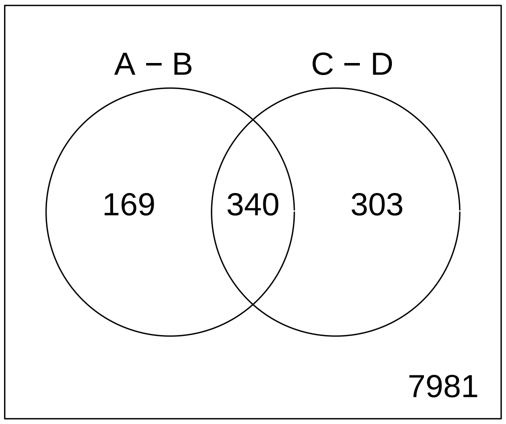

Figure 1: Venn diagram

```
[1] "Symbol"      "Description" "GO"          "GOID"        "Chromosome"
[6] "ChromLoc"

> mart <- useMart("ensembl", "hsapiens_gene_ensembl")

Checking attributes and filters ... ok

> annBM(mart)

[1] "Symbol"      "Description" "GO"          "GOID"        "Chromosome"
[6] "ChromLoc"

> mart <- useMart("ensembl", "celegans_gene_ensembl")

Checking attributes and filters ... ok

> annBM(mart)

[1] "Description" "GO"          "GOID"        "Chromosome"

> martDisconnect(mart)
```