Extracting affy and limma objects from affylmGUI files

James Wettenhall

April 22, 2004

This vignette gives a short example showing how to extract affy and limma data objects from files saved by affylmGUI. This could be used for advanced limma analysis by an expert user after some preliminary analysis with affylmGUI by someone unfamiliar with the command-line interface.
We will use a file `EstrogenContrastsComputed.lma` which unfortunately is not available for downloading because it is too large. Efforts will be made in the future to provide a smaller data set as an example.
The `.lma` extension used by affylmGUI is simply a three-letter abbreviation of limma (Linear Models for Microarrays). This file is in fact a standard `.RData` file and can be loaded into any R session as described below.

```
> load("EstrogenContrastsComputed.lma")
```

Firstly, let's load both the affy and limma packages, so that R knows how to display objects defined by Biobase, affy and limma classes (e.g. `AffyBatch` and `MArrayLM`).

```
> library(affy)
> library(limma)
```

Now let's have a look at the R objects available to us:

```
> ls()
```

Now let's look at the RNA targets for this dataset:

```
> Targets
      Name     FileName       Target
1  Abs10.1  low10-1.cel  EstAbsent10
2  Abs10.2  low10-2.cel  EstAbsent10
3 Pres10.1 high10-1.cel EstPresent10
4 Pres10.2 high10-2.cel EstPresent10
```

```
5  Abs48.1  low48-1.cel  EstAbsent48
6  Abs48.2  low48-2.cel  EstAbsent48
7 Pres48.1 high48-1.cel EstPresent48
8 Pres48.2 high48-2.cel EstPresent48
```

Now let's look at the first 10 gene names for this dataset (assuming that the affylmGUI analysis which saved this file had previously found an appropriate annotation package for this chip) :

```
> geneNames[1:10]
 [1] " Rab geranylgeranyltransferase, alpha subunit"
 [2] " mitogen-activated protein kinase 3"
 [3] " tyrosine kinase with immunoglobulin and epidermal growth factor homology doma
 [4] " cytochrome P450, family 2, subfamily C, polypeptide 19"
 [5] " Burkitt lymphoma receptor 1, GTP binding protein (chemokine (C-X-C motif) rec
 [6] " Burkitt lymphoma receptor 1, GTP binding protein (chemokine (C-X-C motif) rec
 [7] " dual specificity phosphatase 1"
 [8] " matrix metalloproteinase 10 (stromelysin 2)"
 [9] " discoidin domain receptor family, member 1"
[10] " protein kinase, interferon-inducible double stranded RNA dependent"
```

The raw probe-level data is stored an an `AffyBatch` object called `RawAffyData`. By typing the name of this object at the prompt, we get a brief summary of this data, including the name of the chip name, the number of samples (8 in this case) and the number of genes (12625 in this case).

```
> RawAffyData
AffyBatch object
size of arrays=640x640 features (25604 kb)
cdf=HG_U95Av2 (12625 affyids)
number of samples=8
number of genes=12625
annotation=hgu95av2
```

We can now obtain the probe set IDs as follows:

```
> cdfenv <- getCdfInfo(RawAffyData)
> probeSetIDs <- ls(cdfenv)
> as.matrix(probeSetIDs[1:10])
      [,1]
 [1,] "100_g_at"
 [2,] "1000_at"
 [3,] "1001_at"
```

```
 [4,] "1002_f_at"
 [5,] "1003_s_at"
 [6,] "1004_at"
 [7,] "1005_at"
 [8,] "1006_at"
 [9,] "1007_s_at"
[10,] "1008_f_at"
```

Now let's check whether normalized affy data is available by checking the Boolean variable `NormalizedAffyData.Available`.

```
> NormalizedAffyData.Available
TRUE
```

Now let's check what method was used to normalize the data. Currently there are two choices: RMA (Robust Multiarray Averaging) and PLM (Probe-level Linear Models).

```
> NormMethod
[1] "RMA"
```

Since the normalized data is available, set's see a summary of it, by typing the name of the expression set object storing the normalized data, `NormalizedAffyData`.

```
> NormalizedAffyData
Expression Set (exprSet) with
        12625 genes
        8 samples
                 phenoData object with 1 variables and 8 cases
         varLabels
                sample: arbitrary numbering
```

Now let's see the design matrix used for linear modelling (which is automatically determined from the Targets file).

```
> design
             EstAbsent10 EstAbsent48 EstPresent10 EstPresent48
low10-1.cel            1           0            0            0
low10-2.cel            1           0            0            0
high10-1.cel           0           0            1            0
high10-2.cel           0           0            1            0
low48-1.cel            0           1            0            0
low48-2.cel            0           1            0            0
high48-1.cel           0           0            0            1
high48-2.cel           0           0            0            1
```

Now let's see how many contrast parameterizations have been defined (i.e. how many contrast matrices).

```
> NumContrastParameterizations
[1] 1
```

In this case, there is only one contrast parameterization. Now let's have a look at the objects stored within this contrast parameterization. The '1' in double square-brackets represents the first (and only) contrast parameterization.

```
> names(ContrastParameterizationList[[1]])
[1] "NumContrastParameterizations"     "fit"
[3] "eb"                               "contrastsMatrixInList"
[5] "ContrastParameterizationNameText"
```

There is an object called contrastsMatrixInList, which is a list object containing the contrast matrix, and some information about how the user created that contrast matrix, in this case by requesting several pariwise comparisons between the four cases in the 2X2 factorial design (estrogen abs/pres X time 10/40) rather than manually entering the contrast matrix numerically.

```
> ContrastParameterizationList[[1]]$contrastsMatrixInList
$contrasts
             (EstPresent10)-(EstAbsent10) (EstPresent48)-(EstAbsent48) (EstPresent48
EstAbsent10                           -1                            0
EstAbsent48                            0                           -1
EstPresent10                           1                            0
EstPresent48                           0                            1

$contrastsCreatedFromDropDowns
[1] TRUE

$Param1
[1] 3 4 4

$Param2
[1] 1 2 3
```

Now let's look at the linear model fit object. Currently, `affylmGUI` still uses the old `lm.series` from `limma` rather than `lmFit`, so the `fit` object is just a standard R list object, so typing `ContrastParameterizationList[[1]]$fit` and pressing enter would

display a lot more data than you would generally want to see scrolling past your screen. Soon this will probably change, so that `lmFit` will be used to create an fit object of class `MArrayLM`. In either case, the components of the fit object can be viewed as follows:

```
> names(ContrastParameterizationList[[1]]$fit)
[1] "coefficients"   "stdev.unscaled" "sigma"          "df.residual"    "contrasts"
[6] "Amean"
```

Empirical bayes statistics can be obtained from the `"eb"` component of `ParameterizationList[[1]]`. Note that recent versions of limma encourage users to calculate empirical bayes statistics using `eBayes`, rather than `ebayes`, whereas at the time of writing limmaGUI still uses the old `ebayes` method, which produces a standard list object, meaning that typing `ParameterizationList[[1]]$eb` and pressing enter will display all the data in the list, rather than a summary. The components of the empirical bayes list object can be viewed as follows:

```
> names(ContrastParameterizationList[[1]]$eb)
 [1] "coefficients"   "stdev.unscaled" "sigma"          "df.residual"    "contrasts"
 [6] "df.prior"       "s2.prior"       "var.prior"      "proportion"     "s2.post"
[11] "t"              "p.value"        "lods"
```

For example, the moderated t statistics can be obtained as follows:

```
> ContrastParameterizationList[[1]]$eb$t
```