

# Building R objects from ArrayExpress datasets

Audrey Kauffmann

April 14, 2009

## 1 ArrayExpress database

ArrayExpress is a public repository for transcriptomics and related data, which is aimed at storing MIAME-compliant data in accordance with MGED recommendations. The ArrayExpress Data Warehouse stores gene-indexed expression profiles and related measurements from a curated subset of experiments in the repository. Currently, 100,000 hybridizations are represented in ArrayExpress. Please visit <http://www.ebi.ac.uk/arrayexpress/> for more information on the database.

## 2 MAGE-TAB format

In the repository, for each dataset, ArrayExpress stores a MAGE-TAB document with standardized format. A MAGE-TAB document contains four different types of files Investigation Description Format (IDF), Array Design Format (ADF), Sample and Data Relationship Format (SDRF), and Raw and processed data files. The tab-delimited IDF file contains top level information about the experiment including title, description, submitter contact details and protocols. The tab-delimited ADF file describes the design of an array, e.g., what sequence is located at each position on an array and what the annotation of this sequence is. The tab-delimited SDRF file contains the sample annotation and the relationship between arrays as provided by the submitter. The raw zip file contains the raw files (like the CEL files for Affymetrix chips or the GPR files from the GenePix scanner for instance), and the processed zip file contains all processed data in one generic tab delimited text format.

## 3 Querying the database

With the `queryAE` function, you can query the ArrayExpress repository. Two arguments are available, 'keywords' and 'species'. You can use both of them simultaneously or each of them independently, depending on your needs. If you want to use several words, they need to be separated by a '+' without any space. Here is an example where the object `sets` contains the identifiers of all the datasets for which the word 'leukemia' was found in the description and for which the Homo sapiens is the studied species. You need to be connected to Internet to have access to the database.

```
> library("ArrayExpress")
> sets = queryAE(keywords = "leukemia", species = "homo+sapiens")
```

In October 2008, this query was giving 167 identifiers. The output is a dataframe with the identifiers and two columns giving the availability of the raw and processed data. When raw and processed data are available for the same dataset, a unique identifier is given for both. There is no way to distinguish between a processed and a raw dataset just from the identifier. The column 'Raw' from the output of `ArrayExpress` allows to know if the raw data are available for a data set. For a more extended querying mode, the ArrayExpress website offers an advanced query interface with detailed results.

## 4 Import an ArrayExpress dataset into R

### 4.1 Call of the function `ArrayExpress`

Once you know which identifier you wish to retrieve, the function `ArrayExpress` can be called, using the following arguments:

- *input*: an ArrayExpress identifier for which the raw data are available.
- *path*: the name of the directory in which the files downloaded from the ArrayExpress repository will be extracted. The default is the current directory.
- *save*: if TRUE, the files downloaded from the database will not be deleted from 'path' after executing the function.
- *rawcol*: by default, the columns are automatically selected according to the scanner type. If the scanner is unknown or if the user wants to use different columns than the default, the argument 'rawcol' can be set. For two colour arrays it must be a list with the fields 'R', 'G', 'Rb' and 'Gb' giving the column names to be used for red and green foreground and background. For one colour arrays, it must be a character string with the column name to be used. These column names must correspond to existing column names of the expression files.

You still need to be connected to Internet to have access to the database.

### 4.2 Examples and output format

**Simple example** The output object is an *AffyBatch* if the ArrayExpress identifier corresponds to an Affymetrix experiment, it is an *ExpressionSet* if the identifier corresponds to a one colour non Affymetrix experiment and it is a *NChannelSet* if the identifier corresponds to a two colours experiment.

```
> rawset = ArrayExpress("E-MEXP-1422")
```

In this example, 'E-MEXP-1422' being an Affymetrix experiment, 'rawset' is an *AffyBatch*. The expression values of 'rawset' are the content from the CEL files. The *phenoData* of 'rawset' contains the whole sample annotation file content from the MAGE-TAB sdrf file. The *featureData* of 'rawset' contains the feature annotation file content from the MAGE-TAB adf file. The *experimentData* of 'rawset' contains the experiment annotation file content from the MAGE-TAB idf file.

**Example when the column names are needed** In the case of non Affymetrix experiments, `ArrayExpress` decides automatically, based on known quantitation types, which column from the scan files are to be used to fill the *exprs*. However, in some cases the scanner software is not recognized or unknown and this decision cannot be made automatically. The argument 'rawcol' is then needed. Here is an example.

```
> eset = try(ArrayExpress("E-DKFZ-1"))
```

Here, the object cannot be built because the columns are not recognized. The error message also gives a list of possible column names. We can then call the function again, feeding the argument 'rawcol' with the appropriate column names.

```
> eset = ArrayExpress("E-DKFZ-1",  
+   rawcol = list(R = "Software Unknown:Foreground red",  
+   G = "Software Unknown:Foreground green",  
+   Rb = "Software Unknown:Background red" ,  
+   Gb = "Software Unknown:Background green"))
```

Then `eset` is created. However, there is still a warning, the *phenoData* cannot be built. This means that the object is correctly created but the sample annotation has not been attached to it. It is still possible to manually correct the files and try to build the object. To do so, the functions `getAE` and `magetab2bioc`, used by the `ArrayExpress` function, can be called separately.

## 5 Download an ArrayExpress dataset on your local machine

It is possible to only download the data, by calling the function `getAE`. The arguments `'input'`, `'path'`, `'save'` are the same than for the `ArrayExpress` function. The argument `'type'` determines if you retrieve the MAGE-TAB files with the raw data only (by setting `'type'` to `'raw'`), the MAGE-TAB files with the processed data only (by setting `'type'` to `'processed'`) or if you retrieve all the MAGE-TAB files, both raw and processed (by setting `'type'` to `'full'`). Here, you also need Internet connection to access the database.

```
> full = getAE("E-MEXP-1422", type = "full")
```

Here, the output is a list of all the files that have been downloaded and extracted in the directory `'path'`.

## 6 Build an R object from local MAGE-TAB

If you have your own raw MAGE-TAB data or if you want to build an R object from existing MAGE-TAB data that are stored locally on your computer. The function `magetab2bioc` can convert them into an R object.

The arguments for the `magetab2bioc` are:

- *rawfiles* all the expression files to use to create the object. The content of the raw.zip MAGE-TAB file.
- *sdrf* the sdrf file from MAGE-TAB.
- *adf* the adf file from MAGE-TAB.
- *idf* the idf file from MAGE-TAB.
- *path* see the `ArrayExpress` function arguments.
- *rawcol* see the `ArrayExpress` function arguments.
- *save* see the `ArrayExpress` function arguments.

As an example, we can use the files that we have downloaded in the previous example.

```
> rawset= magetab2bioc(rawfiles = full$rawfiles,
+   sdrf = full$sdrf,
+   adf = full$adf,
+   idf = full$idf)
```

The object `rawset` is an *AffyBatch*.

## 7 ArrayExpress package efficiency

For this first release, the *ArrayExpress* package works well on 96% of the raw data available on the database. Ongoing work aims at developing a function to deal with the processed data as well.