Building R objects from ArrayExpress datasets

Audrey Kauffmann

May 3, 2016

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 Gene Expression Atlas stores gene-condition summary expression profiles from a curated subset of experiments in the repository. Currently, around 917 460 assays 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 five different types of files Investigation Description Format (IDF), Array Design Format (ADF), Sample and Data Relationship Format (SDRF), the raw data files and the 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

It is possible to query the ArrayExpress repository, using the `queryAE` function. Two arguments are available, 'keywords' and 'species'. You can use both of them simultaneously or each of them independently, according to 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 'pneumonia' 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 = "pneumonia", species = "homo+sapiens")
```

In August 2012, this query retrieved 21 identifiers. The output is a dataframe with the identifiers and 7 columns giving the number of data raw and processed. 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 `queryAE` allows to know if the raw data are available for a data set. The following columns contain the release date of the dataset on the database, the pubmed ID of the publication related to the dataset, the species, the experiment design and the experimental factors. The

experiment design defines what was studied by the authors, such as time series, disease state or compound treatment. The experimental factors are the values of the studied factors, "Day 1", "Day 2" for instance. 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:

- *accession*: an ArrayExpress experiment 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.
- *dataCols*: 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 'dataCols' 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 ouput 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")

Reading in : /tmp/RtmpOkitOC/AF15.CEL
Reading in : /tmp/RtmpOkitOC/AF16.CEL
Reading in : /tmp/RtmpOkitOC/AF6.CEL
Reading in : /tmp/RtmpOkitOC/AF14.CEL
Reading in : /tmp/RtmpOkitOC/AF7.CEL
Reading in : /tmp/RtmpOkitOC/AF8.CEL
```

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 whole 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-MEXP-1870"))
```

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 'dataCols' with the appropriate column names.

```
> eset = ArrayExpress("E-MEXP-1870",
+   dataCols=list(R="ScanArray Express:F650 Mean",
+     G="ScanArray Express:F550 Mean",
+     Rb="ScanArray Express:B650 Mean",
+     Gb="ScanArray Express:B550 Mean"))
```

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 `ae2bioc`, 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' and 'path' 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.

```
> mexp1422 = 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 raw 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 `ae2bioc` can convert them into an R object.

The arguments for the `ae2bioc` are:

- *mageFiles* A list as given from `getAE` function. Containing the elements 'rawFiles' (the expression files to use to create the object), 'sdrf' (name of the sdrf file), 'idf' (name of the idf file), 'adf' (name of the adf file) and 'path' (the name of the directory containing these files).
- *dataCols* by default, the columns are automatically selected according to the scanner type. If the scanner is unknown, the argument 'dataCols' 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.

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

```
> rawset= ae2bioc(mageFiles = mexp1422)
```

```
Reading in : /tmp/Rtmp9rZcCZ/Rbuild182e1a615d86/ArrayExpress/vignettes/AF15.CEL
Reading in : /tmp/Rtmp9rZcCZ/Rbuild182e1a615d86/ArrayExpress/vignettes/AF16.CEL
Reading in : /tmp/Rtmp9rZcCZ/Rbuild182e1a615d86/ArrayExpress/vignettes/AF6.CEL
Reading in : /tmp/Rtmp9rZcCZ/Rbuild182e1a615d86/ArrayExpress/vignettes/AF14.CEL
Reading in : /tmp/Rtmp9rZcCZ/Rbuild182e1a615d86/ArrayExpress/vignettes/AF7.CEL
Reading in : /tmp/Rtmp9rZcCZ/Rbuild182e1a615d86/ArrayExpress/vignettes/AF8.CEL
```

The object `rawset` is an *AffyBatch*.

7 Build an R object from local processed MAGE-TAB

Processed data in the database are less uniform as processing methods vary a lot. To import a processed dataset from ArrayExpress, three steps are required: (i) download the dataset using `getAE`, (ii) identify which column is of interest thanks to `getcolproc`, (iii) create the R object with `procset`.

7.1 Identification of the columns to extract

Once the data are downloaded, we need to know the different column names available in the processed file to be able to choose a relevant one to extract. The function `getcolproc` needs, as an input, a list containing two slots:

- *procfile* the name of the processed expression file.
- *path* the name of the directory containing the 'procfile'.

This kind of list is given as an output from the function `getAE`.

```
> cn = getcolproc(mexp1422)
> show(cn)
```

```
[1] "Composite Element REF"   "Software Unknown:GC-RMA"
[3] "Unknown:Detection Call"
```

`cn` is a character vector with the available column names in the 'procfile'.

7.2 Creation of the object

We can now create the *ExpressionSet* using `procset`. This function has two arguments:

- *files* a list as given by `getAE` containing the names of the processed, sdrf, idf, adf files and the path.
- *procol* the name of the column chosen after using `getcolproc`.

```
> proset = procset(mexp1422, cn[2])
```

```
Reading processed data matrix file,  /tmp/Rtmp9rZcCZ/Rbuild182e1a615d86/ArrayExpress/vignettes/ex
```

`proset` is an *ExpressionSet* containing the processed log(ratio) of the dataset E-MEXP-1422.

8 Example of a standard microarray analysis using data from ArrayExpress

In this section, we are briefly describing an analysis from the data import to the identification of differentially expressed genes, of data publicly available on ArrayExpress.

The first step consists of importing a dataset from ArrayExpress. We chose E-MEXP-1416. This dataset studies the transcription profiling of melanized dopamine neurons isolated from male and female patients with Parkinson disease to investigate gender differences.

```
> AEset = ArrayExpress("E-MEXP-1416")
```

As AEset is an Affymetrix experiment, we can use RMA normalisation to process the raw data, please read the rma function help.

```
> library("affy")
> AEsetnorm = rma(AEset)
```

To check the normalisation efficiency, we can run a quality assessment. For details on the arguments used, please read the arrayQualityMetrics vignette. Please note that, at the time of its release (Oct 2010), Bioconductor 2.7 didn't include a 64-bit Windows binary version of the *arrayQualityMetrics* package (but this might change in the future).

```
> fac = grep("Factor.Value",colnames(pData(AEsetnorm)), value=T)

> if (suppressWarnings(require("arrayQualityMetrics", quietly=TRUE))) {
+   qanorm = arrayQualityMetrics(AEsetnorm,
+     outdir = "QAnorm",
+     intgroup = fac)
+ }
```

Now that we have ensured that the data are well processed, we can search for differentially expressed genes using the package *limma*. To understand the details of each steps, please see the limma user guide.

```
> library("limma")
> facs =  pData(AEsetnorm)[,fac]
> facs[facs[,2]=="female",2]="F"
> facs[facs[,2]=="male",2]="M"
> facs[facs[,1]=="Parkinson disease",1]="parkinson"
> facs = paste(facs[,1],facs[,2], sep=".")
> f = factor(facs)
> design = model.matrix(~0+f)
> colnames(design) = levels(f)
> fit = lmFit(AEsetnorm, design)
> cont.matrix = makeContrasts(normal.FvsM = normal.F-normal.M,
+     parkinson.FvsM = parkinson.F-parkinson.M,
+     Diff=(parkinson.F-parkinson.M)-(normal.F-normal.M),
+     levels=design)
> fit2 = contrasts.fit(fit, cont.matrix)
> fit2 = eBayes(fit2)
> res = topTable(fit2, coef = "parkinson.FvsM", adjust = "BH")
```

Here we end up with a list of genes that are differentially expressed between the female and the male patients having Parkinson's disease, which was the topic of the paper, but one can perform other comparisons with the same data.