Description of the *ArrayExpressDataManage* package: Data Management of ArrayExpress Experiments at Local File System

Markus Schmidberger [*][†] Ulrich Mansmann[*]

August 7, 2009

Contents

[*]Division of Biometrics and Bioinformatics, IBE, University of Munich, 81377 Munich, Germany

[†]Package maintainer, Email: `schmidb@ibe.med.uni-muenchen.de`

1 Abstract

The *ArrayExpressDataManage* package is part of the Bioconductor[1] [2] project. The package extends the *ArrayExpress* package. The *ArrayExpress*[5] package is an interface to the ArrayExpress (AE) Repository at EBI and builds Bioconductor data structures: `ExpressionSet`, `AffyBatch`, `NChannelSet`. For more details see the *ArrayExpress* vignettes.

The *ArrayExpressDataManage* package offers data management of AE experiments at the local file system and uses a optimized data structure. Thereby the data set can be reused from anyone without providing the whole data set, e.g., as new experiment in AE. The dataset can be regenerated very simply from the public available experiments in AE and further experiments can be added to the data structure. The main function of the package is the function `dapply()`, which is an apply-like function to apply a function FUN to files in a directory structure.

This is a hands-on introduction to the functionality of the *ArrayExpress-DataManage* package and uses a standard microarray analysis procedure to describe the functionality. For more details about a standard analysis process for microarray data see for example [1].

```
R> library(ArrayExpressDataManage)
```

2 Changes to previous Versions

For major changes see the NEWS file in the source code of the package or use the function `readNEWS()`.

3 Directory Structure at Local File System

The *ArrayExpressDataManage* package is based on the optimized local file system data structure. To keep a clear data management, the raw data and processed data are saved in a defined directory structure on the hard disk. In this example the directory structure for a large cancer study is described:

large-cancer-study Top directory for the large cancer study.

[1] `http://www.bioconductor.org/`

Cancer-entity-XYZ Every cancer entity has its own directory. All experiments belonging to this entity will be stored in this directory.

- **Array.adf.txt** Array design description.
- **Experiment-XYZ.raw.gz** Packed CEL files for one experiment.
- **Experiment-XYZ.sdrf.txt** Detailed sample and data relationship annotation file for one experiment.
- **Experiment-XYZ.idrf.txt** Investigation description for one experiment.
- **Experiment-XYZ_eset.Rdata** *R* objects of preprocessed experiment data saved as binary Rdata file.

Cancer-entity-XYZ_eset.Rdata *R* objects of all experiment data, belonging to cancer entity XYZ, preprocessed together and saved as binary Rdata file.

complete_eset.Rdata *R* objects of all experiment data preprocessed together and saved as binary Rdata file.

This file structure is optimized for the data processing with the *R* language and for re-usability of intermediate results. For example, the preprocessed data are stored in the data structure or further results, e.g., graphs, can be stored, too. The CEL files are only stored as packed files to save disk memory. This data structure can be reused for other study types. The grouping into cancer entities can be generalized to all other kinds of groups (for example age or sex).

This simple example creates the data structure and downloads the two experiments "E-GEOD-8003" and "E-GEOD-10097" from the AE database. These experiments are assigned to the group "example".

```
R> dir <- tempdir()
R> data <- list(example=c("E-GEOD-8003", "E-GEOD-10097"))
R> path <- createDataStruct(path = dir, data = data,
+                    name = "ArrayExpressData")
```

```
Group:  example
        E-GEOD-8003
        E-GEOD-10097
```

The `createDataStruct` creates a character object with the path location. This object is required in most of the following functions.

4 The main Function: dapply()

The main function of the new package is the `dapply()` function.

```
R> res <- dapply(function(x) basename(x),
+                   path=path, pattern=".txt", recursive=TRUE)
R> unlist(res)

[1] "A-AFFY-33.adf.txt"     "E-GEOD-10097.idf.txt"  "E-GEOD-10097.sdrf.txt"
[4] "E-GEOD-8003.idf.txt"   "E-GEOD-8003.sdrf.txt"
```

This is an apply-like function which returns a list of values obtained by applying a function `FUN` to files in a directory. The directory can be defined by the variable `path` and the file type with a regular expression in the parameter `pattern`.

5 A Standard Analysis Process for Microarray Data

Further functionality of the *ArrayExpressDataManage* package is described with a standard microarray analysis procedure. For more details about a standard analysis process for microarray data see for example [1].

5.1 Statistic and Overview for the Experiments in the Data Structure

There are two useful functions to create some statistic about the experiments in the data structure and an overview table about the experiments (see Table 1):

```
R> create_statistic(path = path)

        Experiments Arrays ChipType
example           2      5        5

R> info <- create_overview(path=path)
R> require(xtable)
R> xinfo <- xtable(info,
+          caption="Small selection of selected AE experiments",
+          label="tab:experiment_overview_small",
+          align="lllp{4cm}p{9cm}ll")
R> print(xinfo, type="latex",
+          include.rownames=FALSE, append=FALSE,
+          latex.environments=c("tiny","center"),
+          floating.environment="sidewaystable",
+          table.placement = "htp")
```

5.2 Check for Duplicates

Using a lot of experiments from the AE database, there can be duplicate arrays in different experiments. These arrays have to be removed from the analysis to omit unbalanced high-level analysis

```
R> duplicateList <- checkForDuplicates(path = path)

        Extract: ..

R> duplicateList

character(0)
```

A list with duplicate arrays is returned, which can be manually removed or ignored in the preprocessing. In this case there are no duplicates in the example data.

5.3 Quality Assessment & Control

As next step the quality of the arrays should be assessed and controlled. For more details see for example [1, 3, 6]. A very useful package for the quality control is *arrayQualityMetrics* package[6]. This package can be used for the quality control of the experiments in the data structure.

A function with the selected quality assessment method(s) has to be defined and the list of outliers must be the return values. Then the function `qaComplete()` for the complete quality control can be used.

ID	Entity	Title	Description	Arrays	PubMed ID
E-GEOD-10097	example	Transcript profiling of oestrogen treatment of primary human neuronal and glial cell cultures	The purpose of this experiment was to identify oestrogen regulated genes in human primary cell cultures of neuronal and glial cells modelling the developing human nervous system. We were especially interested in genes involved in proliferation, differentiation and migration of neuronal cells and genes involved in or linked to neurodegenerative diseases. We have therefore assessed gene expression c ...	2/2	
E-GEOD-8003	example	CD34 Overexpression	In order to investigate the role of CD34 antigen in haematopoietic commitment, we overexpressed the human CD34 cDNA in human CD34+ cells by retroviral gene transfer. Experiment Overall Design: In a set of 10 independent experiments, gene transfer efficiency, assessed by flow cytometry analysis of {\~A}?{\^A}?LNGFR positivity, ranged 15 to 30%. Transduced cells were than purified for NGFR expression at day ...	3/3	

Table 1: Small selection of selected ArrayExpress experiments.

```
R> qaFunBox <- function(x)
+  {
+          library(arrayQualityMetrics)
+          obj <- x
+          prepdata<- aqm.prepdata(expressionset = obj, do.logtransform=T)
+          bo<- aqm.boxplot(obj, dataprep=prepdata)
+          return(bo$outliers$mean)
+  }
R> qa <- qaComplete(path = path, qaFun = qaFunBox)

QA complete :
        Extract: ..

        QA
         0  Arrays of wrong Chip Types deleted.
        Remove extracted data on master

R> qa$out

numeric(0)
```

The output is a list file with serveral infromation. The names of the arrays with bad quality are stored in the `out` slot. In this case there is no array with bad quality.

5.4 Preprocessing of Microarray Data

The next steps is the preprocessing of the raw data. It can be interesting to preprocess all data together, to preprocess only data from one group or to preprocess all experiment seperate. With the parameter `preprocessFUN` the favored (complete) preprocessing function can be selected, e.g. rma, vsn or rmaPara from the *affyPara*[7] package.

```
R> library(affy)
R> esetC <- preprocessComplete(path = path,
+                    preprocessFUN = rma)

Process complete :
        Extract: ..
        Preprocess
Background correcting
Normalizing
Calculating Expression
        Remove extracted data
```

```
R> esetC
```

```
[1] TRUE
```

```
R> esetG <- preprocessGroups(path = path,
+                    preprocessFUN = rma)
```

```
R> esetE <- preprocessExperiments(path = path,
+                    preprocessFUN = rma)
```

In this case the complete preprocessing and the preprocessing of the data from one group it the same, because there is only one group.

As you can see, the results are stored in the local file structure:

```
R> res <- dapply(function(x) basename(x), path=path,
+           pattern=".Rdata", recursive=TRUE)
R> unlist(res)
```

```
[1] "complete_eset.Rdata" "complete_qa.Rdata"
```

5.5 Correct Batch Effect

Combining experiments from different labs in most cases involves a so called batch effect. This non-biological experimental variation is commonly observed across multiple batches of microarray experiments. For example, the a heatmap grafic can be used to visualize the batch effect. Different methods habe been proposed to filter batch effects from data. All of them have different advantages and drawbacks. This package provides a parametric and non-parametric empirical Bayes framework called 'Combatting batch effects when combining batches of gene expression microarray data' (ComBat) and developed from [4]. It is robust to outliers in small (<10) sample sizes and performs comparable to existing methods for large samples. It is the only algorithm correcting batch effects, which is available in

R code (`http://statistics.byu.edu/johnson/ComBat`). Small changes of the code were required for the application to the latest Bioconductor `ExpressionSet` object.

```
R> eset_batch <- correctBatch(path = path)
```

Again, the results are stored in the local file structure and can be reused.

5.6 Nonspecific Filtering

Nonspecific filtering is a very common procedure after preprocessing and before high-level analysis (see [3]). In this step the genes are filter for different criterias and only the intersting genes are used for further analysis.

```
R> filterExperiments(path = path)

Filter complete :
         15825 probes removed.
```

Again, the results are stored in the local file structure and can be reused.

5.7 Some High-Level Analysis

There are a lot of different high-level analysis tools, which can be applied to the preprocessed microarray data. This example demonstrates the analysis of correlation between genes.

```
R> library(pcalg)
R> graphs <- dapply(function(eset){
+          load(eset)
+          g <- pcAlgo( t( exprs(eSet)[50:100,] ), alpha=0.05 )
+          return(g)
+  }, path=path, pattern="_eset.Rdata", recursive=TRUE)
```

6 Clean Data Structure

After a lot of work it could be very useful to clean the data structure from temporary saved results:

```
R> clean(path = path)
```

References

[1] Robert Gentleman, Vincent Carey, Wolfgang Huber, Rafael Irizarry, and Sandrine Dudoit. *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Springer, 1 edition, August 2005.

[2] Robert C. Gentleman, Vincent J. Carey, Douglas M. Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, Kurt Hornik, Torsten Hothorn, Wolfgang Huber, Stefano Iacus, Rafael Irizarry, Friedrich Leisch, Cheng Li, Martin Maechler, Anthony J. Rossini, Gunther Sawitzki, Colin Smith, Gordon Smyth, Luke Tierney, Jean Y. H. Yang, and Jianhua Zhang. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5:R80, 2004.

[3] Florian Hahne, Wolfgang Huber, Robert Gentleman, and Seth Falcon. *Bioconductor Case Studies*. Springer Publishing Company, Incorporated, 2008.

[4] W. Evan Johnson, Cheng Li, and Ariel Rabinovic. Adjusting Batch Effects in Microarray Expression Data using empirical Bayes Methods. *Biostatistics*, 8(1):118–127, Jan 2007.

[5] Audrey Kauffmann. *ArrayExpress: Access the ArrayExpress Microarray Database at EBI and build Bioconductor data structures: ExpressionSet, AffyBatch, NChannelSet*, 2009. R package version 1.4.0.

[6] Audrey Kauffmann, Robert Gentleman, and Wolfgang Huber. *arrayQualityMetrics* – a Bioconductor Package for Quality Assessment of Microarray Data. *Bioinformatics*, 25(3):415–416, Feb 2009.

[7] Markus Schmidberger and Ulrich Mansmann. *affyPara* - a Bioconductor Package for Parallelized Preprocessing Algorithms of Affymetrix Microarray Data. *Bioinformatics and Biology Insights*, 3, 2009.