Documentation of the RMAGEML package

Steffen Durinck[‡,*], Joke Allemeersch[‡,†], Vincent J Carey[¶], Yves Moreau[‡], Bart De Moor[‡]

October 31, 2011

[‡]Department of Electronical Engineering, ESAT-SCD, K.U.Leuven, Kasteelpark Arenberg 10, 3001 Leuven-Heverlee, Belgium, `http://www.esat.kuleuven.ac.be/~dna/BioI`
and [¶]Channing Laboratory, Brigham and Women's Hospital, 75 Francis Street, Boston 02115, USA

Contents

*Steffen.Durinck@esat.kuleuven.ac.be
†Joke.Allemeersch@esat.kuleuven.ac.be

1 Introduction

MAGE-ML or Microarray Gene Expression Markup Language is a language designed to describe and exchange information about microarray experiments. MAGE-ML is based on XML and can describe microarray designs, microarray experiment setups, gene expression data, and data analysis results.
This package provides the link between MAGE-ML files and BioConductor. It gives the possibility to read in MAGE-ML files that describe cDNA microarray experiments. The functions convert the MAGE-ML files into the customary BioConductor objects (i.e., `marrayLayout, marrayInfo` and `marrayRaw` objects or limma `RGList` objects).

Here we give a short introduction to the Microarray and GeneExpression Object Model (MAGE-OM) and how we implemented the extraction of information necessary to make BioConductor objects. For a full description of MAGE-OM, we refer to the Gene Expression Specification: `http://www.omg.org/cgi-bin/doc?formal/03-02-03`.
The main classes of the MAGE object model are BioSequence, Quantitationtype, ArrayDesign, DesignElement, Array, BioMaterial, BioAssay, BioAssayData, Experiment, HigherLevelAnalysis, Protocol, Description, AuditAndSecurity, Measurement, and BioEvent.
In MAGE-ML these translate into packages with the same name. The packages needed for building BioConductor objects are BioAssayData, BioAssay, BioMaterial, BioSequence, ArrayDesign, and DesignElement.
The DesignElement package contains a mapping of *Features*, which are the actual features present on the array, to *Reporters*, the reporter a feature represents. The DesignElement package also provides a mapping from *Reporters* to their corresponding *BioSequence* references. These *BioSequence* objects are characterized by their name and database entries in the BioSequence package. The ArrayDesign package contains information on the layout of the array. From this package, we can derive the position of each *Feature* on the array in terms of *Zone* (block or grid) and row and column within each *Zone*. The BioAssayData package describes the feature references that were assayed and the measured and derived *QuantitationTypes*. The BioAssay package describes the different steps in the microarray experiment. The last package used to make BioConductor objects is the BioMaterial package and describes how a sample is treated to obtain, for example, labeled samples used for hybridization.

2 Prerequisites

The RMAGEML package depends on SJava(>= 0.68) and a Java VM, e.g. j2resdk1.4.0. Other dependencies are as the Java-MAGEstk API and Java Xerces included in the package itself.

3 Getting started

Installing the package. The package can be installed as a normal R package: download the RMAGEML_2.0.4.tar.gz package and under Unix use the command

`R CMD INSTALL RMAGEML_2.1.0.tar.gz`.

The equivalent command for Windows is

`Rcmd INSTALL RMAGEML_2.1.0.zip`.

The package automatically loads the Biobase and marrayInput packages from BioConductor and the SJava libraries, so these should be installed as well.

Starting R. Before starting R one should be aware that the RMAGEML package uses SJava and that SJava requires to set the LD_LIBRARY_PATH environment variable before starting R.
Without setting this variable the package won't work

Loading the package. You can load the package into R by typing

```
> ## load up the library
> library(RMAGEML)
```

4 Import to marray packages

4.1 One step import and creation of an marrayRaw object from MAGE-ML files

In the marray packages of BioConductor the design of an array experiment is typically described by an `marrayLayout` and `marrayInfo` object. The function `importMAGEML` parses all MAGE-ML files present in the directory, which is given as a parameter to the function. From these files it creates an `marrayLayout` object, containing the Layout of one type of microarrays, and an `marrayInfo` object containing the gene names and database entries of the features spotted on the array. The name of the database to which the entries refer, is given in the 'notes' slot of the Gnames object. Next the function will extract the raw data values and output a complete *marrayRaw* object as a result.

The function can be tested on the MEXP-14 dataset. This example is available from ArrayExpress at `http://www.ebi.ac.uk/arrayexpress/`.
If one knows which *DesignElement Dimension, QuantitationType Dimension* and *Quantitation Types* are required, the import function can be used as:

```
> ## create marrayRaw object
> datadir <- system.file("MAGEMLdata", package="RMAGEML")
> raw <- importMAGEML(directory = datadir, package = "marray", arrayID = "A-MEXP-14", .

- Java Virtual Machine is running -
parsing MAGEML files
making Layout and Gnames objects
Reading  am2730miame.txt
Reading  am2731miame.txt
Reading  am2732m.txt
Reading  am2736m.txt
Reading  am2737m.txt
Reading  tm1826m.txt
Reading  tm1827m.txt
Reading  tm1829m.txt
Reading  tm1830m.txt
Reading  tm1831m.txt

> print(raw)

An object of class "marrayRaw"
@maRf
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] 5841 2030 2968   45 1828 1975 2077 1775 2202   841
[2,] 2002 1312  421   96  399  557  295  748  465    83
[3,] 2254 2057 1097 1163  649  917  755 1276  985   335
[4,] 2212 1492  782  767  709 1114  620 1004  860   488
[5,]   73   76   42   54   45   49   49   46   47    43
955 more rows ...

@maGf
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]  852  750 1587  135 1625 1183 1598 1108  807  1746
[2,]  652  529  404  162  397  515  285  390  291   190
[3,]  576  615  634  386  734  820  696  573  457   572
[4,]  781  589  733  366  758  848  667  597  559   716
[5,]  157  143  111  124  130  148  135  137  146   131
955 more rows ...

@maRb
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]   42   42   36   39   41   47   40   45   49    42
[2,]   41   41   35   39   40   45   39   43   42    41
```

```
[3,]   41   42   34   41   40   44   40   43   42    42
[4,]   41   40   34   41   40   43   39   43   41    41
[5,]   41   39   34   40   40   42   39   43   41    41
955 more rows ...

@maGb
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]  150  130   87  120  104  135  117  137  168   127
[2,]  147  128   88  111  105  136  116  131  137   121
[3,]  140  124   85  106  105  133  116  128  135   121
[4,]  138  122   88  108  106  133  116  128  134   120
[5,]  138  122   87  106  104  133  114  128  133   118
955 more rows ...

@maW
<0 x 0 matrix>

@maLayout
An object of class "marrayLayout"
@maNgr
[1] 4

@maNgc
[1] 4

@maNsr
[1] 10

@maNsc
[1] 6

@maNspots
[1] 960

@maSub
[1] TRUE

@maPlate
factor(0)
Levels:

@maControls
```

```
factor(0)
Levels:

@maNotes
[1] ""


@maGnames
An object of class "marrayInfo"
@maLabels
[1] "none" "none" "none" "none" "none"
955 more elements ...

@maInfo
[1] aj508733 V00618   aj291984 aj306233 aj310439
142 Levels: V00618 af025843 af034412 af135499 aj132353 aj291832 ... y17187
955 more rows ...

@maNotes
[1] "Identifiers refer to database: DB:embl"


@maTargets
An object of class "marrayInfo"
@maLabels
 [1] "am2730miame.txt" "am2731miame.txt" "am2732m.txt"     "am2736m.txt"
 [5] "am2737m.txt"     "tm1826m.txt"     "tm1827m.txt"     "tm1829m.txt"
 [9] "tm1830m.txt"     "tm1831m.txt"

@maInfo
       Cy3      Cy5
1  AM-Pool AM2730-I
2  AM-Pool AM2731-1
3  AM-Pool AM2732-I
4  AM-Pool AM2736-I
5  AM-Pool AM2737-I
6  AM-Pool TM1826-I
7  AM-Pool TM1827-I
8  AM-Pool TM1829-I
9  AM-Pool TM1830-I
10 AM-Pool TM1831-I
```

```
@maNotes
[1] "Description of the targets"


@maNotes
character(0)
```

If however you do not know which *DesignElement Dimension, QuantitationType Dimension* and *Quantitation Types* to use, you can call the function as follows:

```
> ## create marrayRaw object
> datadir <- system.file("MAGEMLdata", package="RMAGEML")
> if(interactive()){
+ raw <- importMAGEML(directory = datadir, package = "marray")
+ }
```

This will generate a few selection panels which allow selection of the appropriate *DesignElement Dimension, QuantitationType Dimension* and *Quantitation Types*.

4.2 Creation of a Gnames marrayInfo object

If one just wants to make an marrayInfo object containing the gene names and database identifiers of the spotted features the function getGnames can be used.

```
> #To obtain an marrayInfo object containing the database identifiers of the features
> data<-system.file("MAGEMLdata", package="RMAGEML")
> mageom<-importMAGEOM(directory=data)

parsing MAGEML files

> getGnames(mageom, arrayID="A-MEXP-14", DED="DED:707", package = "marray")

An object of class "marrayInfo"
@maLabels
[1] "none" "none" "none" "none" "none"
955 more elements ...

@maInfo
[1] aj508733 V00618   aj291984 aj306233 aj310439
142 Levels: V00618 af025843 af034412 af135499 aj132353 aj291832 ... y17187
955 more rows ...

@maNotes
[1] "Identifiers refer to database: DB:embl"
```

Again leaving out the 'DED' parameter will cause selection panels to pop up displaying the available *DesignElement Dimensions*.

4.3 Creation of an marrayLayout object

In the marray packages the information on the array layout is stored in an marrayLayout object which can be created by the getArrayLayout function.

```
> data<-system.file("MAGEMLdata", package="RMAGEML")
> #To obtain an marrayInfo object containing the database identifiers of the features p
> mageom<-importMAGEOM(directory=data)

parsing MAGEML files

> getArrayLayout(mageom, arrayID="A-MEXP-14", DED="DED:707")

An object of class "marrayLayout"
@maNgr
[1] 4

@maNgc
[1] 4

@maNsr
[1] 10

@maNsc
[1] 6

@maNspots
[1] 960

@maSub
[1] TRUE

@maPlate
factor(0)
Levels:

@maControls
factor(0)
Levels:

@maNotes
[1] ""
```

4.4 Make an marrayRaw object

The function makeMarrayRaw takes a Gnames and Layout object and parameters corresponding to the *DesignElement Dimension*, *QuantitationType Dimension* and *Quantitation Types* to create an *marrayRaw* object.

```
> data<-system.file("MAGEMLdata", package="RMAGEML")
> #To obtain an marrayInfo object containing the database identifiers of the features p
> mageom<-importMAGEOM(directory=data)

parsing MAGEML files

> gnames<-getGnames(mageom, arrayID="A-MEXP-14", DED = "DED:707", package = "marray")
> layout<-getArrayLayout(mageom, arrayID="A-MEXP-14", DED = "DED:707")
> raw <- makeMarrayRaw(mageOM=mageom, layout = layout, gnames = gnames, directory = da

Reading  am2730miame.txt
Reading  am2731miame.txt
Reading  am2732m.txt
Reading  am2736m.txt
Reading  am2737m.txt
Reading  tm1826m.txt
Reading  tm1827m.txt
Reading  tm1829m.txt
Reading  tm1830m.txt
Reading  tm1831m.txt
```

5 Import to limma package

5.1 One step import and creation of a limma RGList object from MAGE-ML files

In the limma package of BioConductor the raw data is stored in an `RGList` object. The function `importMAGEML` parses all MAGE-ML files present in the directory which is given as a parameter to the function. From these files it creates the `RGList` object, containing the layout, gene names and database entries of the features spotted on the array and the foreground and background intensities for the green and red channels.

The function can be tested on the MEXP-14 dataset. This example is available from ArrayExpress at `http://www.ebi.ac.uk/arrayexpress/`.
For import to limma the same function as MAGEML import to marray packages can be used, just adapt the name of the package into limma as follows:

```
> ## create RGList object
> datadir <- system.file("MAGEMLdata", package="RMAGEML")
> raw <- importMAGEML(directory = datadir, package = "limma", arrayID="A-MEXP-14", DED

parsing MAGEML files
Reading  am2730miame.txt
Reading  am2731miame.txt
Reading  am2732m.txt
Reading  am2736m.txt
Reading  am2737m.txt
Reading  tm1826m.txt
Reading  tm1827m.txt
Reading  tm1829m.txt
Reading  tm1830m.txt
Reading  tm1831m.txt

> print(raw)

An object of class "RGList"
$R
     am2730miame.txt am2731miame.txt am2732m.txt am2736m.txt am2737m.txt
[1,]            5841            2030        2968          45        1828
[2,]            2002            1312         421          96         399
[3,]            2254            2057        1097        1163         649
[4,]            2212            1492         782         767         709
[5,]              73              76          42          54          45
     tm1826m.txt tm1827m.txt tm1829m.txt tm1830m.txt tm1831m.txt
[1,]        1975        2077        1775        2202         841
[2,]         557         295         748         465          83
[3,]         917         755        1276         985         335
[4,]        1114         620        1004         860         488
[5,]          49          49          46          47          43
955 more rows ...

$G
     am2730miame.txt am2731miame.txt am2732m.txt am2736m.txt am2737m.txt
[1,]             852             750        1587         135        1625
[2,]             652             529         404         162         397
[3,]             576             615         634         386         734
[4,]             781             589         733         366         758
[5,]             157             143         111         124         130
     tm1826m.txt tm1827m.txt tm1829m.txt tm1830m.txt tm1831m.txt
[1,]        1183        1598        1108         807        1746
```

```
[2,]         515          285          390          291          190
[3,]         820          696          573          457          572
[4,]         848          667          597          559          716
[5,]         148          135          137          146          131
955 more rows ...

$Rb
     am2730miame.txt am2731miame.txt am2732m.txt am2736m.txt am2737m.txt
[1,]              42              42          36          39          41
[2,]              41              41          35          39          40
[3,]              41              42          34          41          40
[4,]              41              40          34          41          40
[5,]              41              39          34          40          40
     tm1826m.txt tm1827m.txt tm1829m.txt tm1830m.txt tm1831m.txt
[1,]          47          40          45          49          42
[2,]          45          39          43          42          41
[3,]          44          40          43          42          42
[4,]          43          39          43          41          41
[5,]          42          39          43          41          41
955 more rows ...

$Gb
     am2730miame.txt am2731miame.txt am2732m.txt am2736m.txt am2737m.txt
[1,]             150             130          87         120         104
[2,]             147             128          88         111         105
[3,]             140             124          85         106         105
[4,]             138             122          88         108         106
[5,]             138             122          87         106         104
     tm1826m.txt tm1827m.txt tm1829m.txt tm1830m.txt tm1831m.txt
[1,]         135         117         137         168         127
[2,]         136         116         131         137         121
[3,]         133         116         128         135         121
[4,]         133         116         128         134         120
[5,]         133         114         128         133         118
955 more rows ...

$genes
  Block Row Column       ID Name
1     1   1      1 aj508733 none
2     1   1      2   V00618 none
3     1   1      3 aj291984 none
4     1   1      4 aj306233 none
```

```
5      1   1       5 aj310439 none
955 more rows ...
```

Similarly if one only specifies the 'directory' and the 'package', selection panels will pop up to select the *DesignElement Dimension, QuantitationType Dimension* and *Quantitation Types*.

5.2 Creating the genes dataframe of an RGList object

In limma the gene names, gene identifiers and layout information is stored in a dataframe which can be created by the getArrayLayoutLimma function.

```
> data<-system.file("MAGEMLdata", package="RMAGEML")
> #To obtain an marrayInfo object containing the database identifiers of the features
> mageom<-importMAGEOM(directory=data)

parsing MAGEML files

> genes<-getArrayLayoutLimma(mageom, arrayID = "A-MEXP-14", DED="DED:707")
> print(genes[1:10,])

   Block Row Column        ID Name
1      1   1      1  aj508733 none
2      1   1      2    V00618 none
3      1   1      3  aj291984 none
4      1   1      4  aj306233 none
5      1   1      5  aj310439 none
6      1   1      6  aj409363 none
7      1   1      7  aj310516 none
8      1   1      8  aj306230 none
9      1   1      9  aj310436 none
10     1   1     10  aj291834 none
```

5.3 Make an RGList object

The function makeRG takes a genes dataframe (containing the layout, gene identifiers and gene names), and parameters corresponding to *DesignElement Dimension, QuantitationType Dimension* and *Quantitation Types* to create a limma `RGList` object.

```
> data<-system.file("MAGEMLdata", package="RMAGEML")
> #To obtain an marrayInfo object containing the database identifiers of the features
> mageom<-importMAGEOM(directory=data)

parsing MAGEML files
```

```
> genes<-getArrayLayoutLimma(mageom, arrayID = "A-MEXP-14", DED = "DED:707")
> raw<- makeRG(mageOM=mageom, genes = genes, directory=data, arrayID="A-MEXP-14", DED

Reading  am2730miame.txt
Reading  am2731miame.txt
Reading  am2732m.txt
Reading  am2736m.txt
Reading  am2737m.txt
Reading  tm1826m.txt
Reading  tm1827m.txt
Reading  tm1829m.txt
Reading  tm1830m.txt
Reading  tm1831m.txt
```