

# Documentation of the MageML package

Joke Allemeersch\*, Steffen Durinck†, Yves Moreau, Bart De Moor

November 25, 2003

Department of Electronical Engineering, ESAT-SCD, K.U.Leuven,  
Kasteelpark Arenberg 10, 3001 Leuven-Heverlee, Belgium,  
<http://www.esat.kuleuven.ac.be/~dna/BioI>

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Getting started</b>	<b>2</b>
<b>3</b>	<b>Creating an <code>marrayLayout</code> and <code>marrayInfo</code> object from MAGE-ML files</b>	<b>3</b>
<b>4</b>	<b>Creating an <code>marrayRaw</code> object</b>	<b>7</b>

## 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).

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>.

---

\*Joke.Allemeersch@esat.kuleuven.ac.be

†Steffen.Durinck@esat.kuleuven.ac.be

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 *BioAssayData* package describes the feature references that were assayed and the quantitation types that are measured. It also provides information on the files in which the raw data are stored.

The *BioSequence* describes known genes or sequences. From its package we extract from each BioSequence the identifier, name, database entry and its corresponding database.

The *ArrayDesign* describes a microarray design that can be printed and hybridized. An array design consists of several features at which reporter sequences are placed. Many features may have the same reporter replicated and a reporter may be specified in one or more array designs. From this package we extract the zone (block or grid) and the row and column of the spotted features within each zone.

The *DesignElement* are the contained classes of the ArrayDesign class and describe through the DesignElements what is intended to be at each location of the array. From this package we extract two mappings, one that maps feature references to reporter references and another that maps reporter references to the corresponding BioSequences.

The *BioAssay* package describes the hybridization events of samples to an array. From this package we extract the PhysicalBioAssay that corresponds to a hybridization and the LabeledExtract references that are involved in this hybridization. The BioMaterial package then gives us more information on the samples: how they were labeled, from which source they came, and so on.

## 2 Getting started

**Installing the package.** The package can be installed as a normal R package: download the `MAGEML_1.0.5.tar.gz` package and under Unix use the command

```
R CMD INSTALL MAGEML_1.0.5.tar.gz.
```

The equivalent command for Windows is

```
Rcmd INSTALL MAGEML_1.0.5.zip.
```

The package automatically loads the BioConductor and the XML library, so these should be installed as well.

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

```
> library(MAGEML)
```

```
Loading required package: marrayClasses
```

```
Welcome to Bioconductor
```

```
Vignettes contain introductory material. To view,  
simply type: openVignette()
```

```
For details on reading vignettes, see  
the openVignette help page.
```

### 3 Creating an `marrayLayout` and `marrayInfo` object from MAGE-ML files

In BioConductor the design of an array experiment is typically described by an `marrayLayout` and `marrayInfo` object. The function `readMageDesign` will parse one or possibly several MAGE-ML files. 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.

The function can be tested on two exemplary data sets. These examples are available from ArrayExpress at <http://www.ebi.ac.uk/arrayexpress/>. The first one, `Iron.xml`, describes a small cDNA microarray experiment. This MAGE-ML file contains among others the `ArrayDesign`, the `BioAssayData`, the `BioSequence` and the `DesignElement` package of the cDNA experiment. These will enable us to construct the `marrayLayout` and `marrayInfo` objects. You can call the function as follows:

```
> datadir <- system.file("data", package = "MAGEML")  
> Design <- readMageDesign(file = "Iron.xml", datadir = datadir)
```

```
Parsing XML documents ... (can take several minutes)
```

```
Making marrayLayout and marrayInfo objects ... (can take several minutes)
```

```
> Layout <- Design$layout  
> Gnames <- Design$info  
> print(Layout)
```

```
Array layout:           Object of class marrayLayout.
```

```
Total number of spots:           1152  
Dimensions of grid matrix:       2 rows by 4 cols  
Dimensions of spot matrices:     18 rows by 8 cols
```

```
Currently working with a subset of 1152 spots.
```

Control spots:

Notes on layout:

Array design is of type ARDES:A-EMBL-1

```
> print(Gnames)
```

Object of class marrayInfo.

	Identification	Gene Name
1	M11147	Human ferritin L chain mRNA, complete cds
2	M11147	Human ferritin L chain mRNA, complete cds
3	M11147	Human ferritin L chain mRNA, complete cds
4	X00351	Human mRNA for beta-actin
5	X00351	Human mRNA for beta-actin
6	X00351	Human mRNA for beta-actin
7	<NA>	T7/T3
8	<NA>	T7/T3
9	<NA>	T7/T3
10	Z11737	H.sapiens mRNA for flavin-containing monooxygenase 4
...		

Number of labels: 0

Dimensions of maInfo matrix: 1152 rows by 2 columns

Notes:

Identification corresponds to accession numbers from following database(s):  
DB:embl

For larger microarray experiments, the ArrayDesign, the BioAssayData, and the DesignElement package are stored in different files. In that case you can specify the files separately. For example the file A-MEXP-5.xml contains the ArrayDesign, the BioSequence, and the DesignElement package of a larger microarray experiment, while the BioAssayData package is stored in E-MEXP-5.xml.

If you have problems in finding out which package is stored in which file, you can use the function `magePackageDetector`. The package contained in the different files, will then appear in your R window.

```
> magePackageDetector(fnames = list("A-MEXP-5.xml", "E-MEXP-5.xml"),  
+   datadir = datadir)
```

File A-MEXP-5.xml contains the following packages:

BioSequence\_package

DesignElement\_package  
ArrayDesign\_package

File E-MEXP-5.xml contains the following packages:

BioMaterial\_package  
BioAssay\_package  
QuantitationType\_package  
BioAssayData\_package

Once you know the different files, you can fill in these files separately as follows.

```
> Design2 <- readMAGEDesign(fArrayDesign = "A-MEXP-5.xml", fDesignElement = "A-MEXP-5.  
+ fBioSequence = "A-MEXP-5.xml", fBioAssayData = "E-MEXP-5.xml",  
+ datadir = datadir)
```

Parsing XML documents ... (can take several minutes)

Making marrayLayout and marrayInfo objects ... (can take several minutes)

```
> Layout2 <- Design2$layout  
> Gnames2 <- Design2$info  
> print(Layout2)
```

Array layout: Object of class marrayLayout.

```
Total number of spots:          5184  
Dimensions of grid matrix:      4 rows by 4 cols  
Dimensions of spot matrices:    18 rows by 18 cols
```

Currently working with a subset of 5184 spots.

Control spots:

Notes on layout:

Array design is of type A-MEXP-7

```
> print(Gnames2)
```

Object of class marrayInfo.

	Identification	Gene	Name
1	AA463173	<NA>	
2	AA463173	<NA>	
3	AA463173	<NA>	

```

4      U70429      <NA>
5      U70429      <NA>
6      U70429      <NA>
7      AW909173    <NA>
8      AW909173    <NA>
9      AW909173    <NA>
10     J04716      <NA>
...

```

Number of labels: 0

Dimensions of maInfo matrix: 5184 rows by 2 columns

Notes:

Identification corresponds to accession numbers from following database(s):

DB:ncbi DB:rzpd DB:image

Depending on to the size of the experiment, this can take some time. Note that you only need to run this function once per microarray design, as the measurements are read in separately in a second function.

It is also possible to use a widget, which permits the user to browse the MAGE-ML files or to enter them in a user-friendly way. Figure 1 shows a snapshot of the widget.

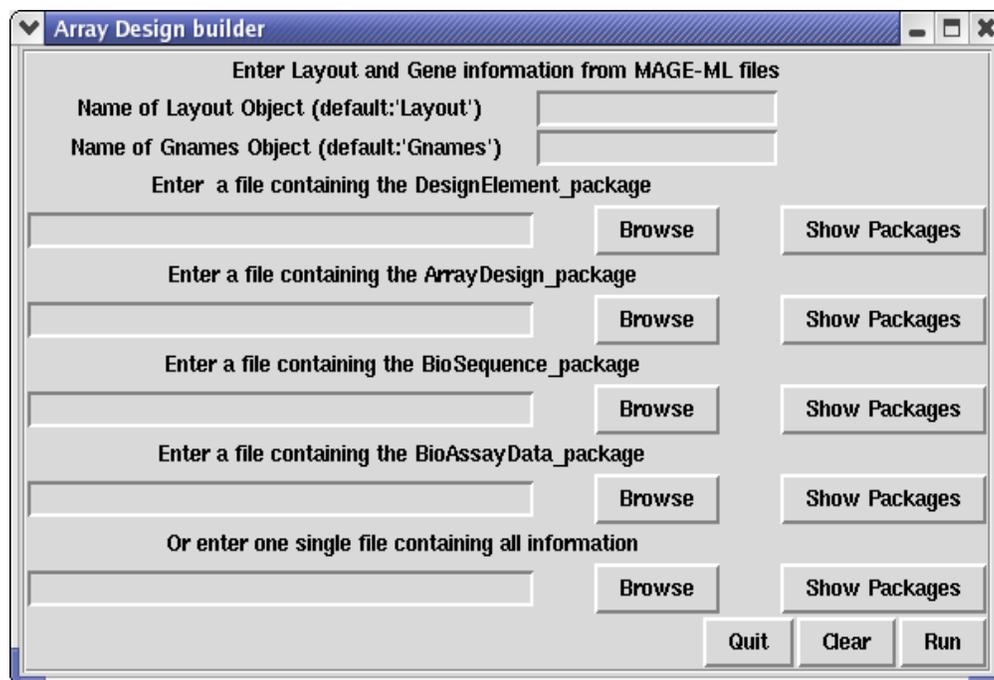


Figure 1: The widget appearing when calling `widget.readMageDesign()`.

You can call this widget with the function

```
> if (interactive()) {  
+   widget.readMageDesign()  
+ }
```

In the first two input boxes, you can choose a name for the Layout and Gnames objects that will be created. By default, the Layout and Gnames objects are called ‘Layout’ and ‘Gnames’, respectively. In the following four input boxes you can specify the different input files. Or, in case all information is contained in one file, you can specify the file just once in the last input box. In case you are not certain about which packages are stored in which file, you can hit the ‘Show Packages’ button. The packages are then printed in the R window. If everything is filled out properly, you just hit the Run button. The proceedings of the program can be followed in the R window.

## 4 Creating an marrayRaw object

Once the `marrayInfo` and `marrayLayout` objects are created, you can start reading in the raw data. These are stored in tab-delimited files, containing the information of one or possibly several hybridizations. The file that includes the `BioAssayData` package also contains the names of the raw data files. Make sure that the XML file and the raw data files are stored in the same directory. Then you can call the function `readMageRaw`.

Take for example again the `Iron.xml` file. In that data set the column with green foreground intensities is called ‘ch1vol’, the red foreground intensities are called ‘ch2vol’. The background intensities are called ‘ch1bgmedian’ and ‘ch2bgmedian’, for green and red respectively. If the names of these intensities are unknown to you, you can obtain a list of possibilities with the function `showQuantitationTypes`. So, for example, you enter

```
> showQuantitationTypes("Iron.xml", datadir = datadir)
```

These are the available quantitation types:

```
flag  
ch1vol  
ch1avvol  
ch1bgarith  
ch1bggeom  
ch1bgmedian  
ch1volbg  
ch1snr  
geoscore  
ch2vol  
ch2avvol  
ch2bgarith  
ch2bggeom
```

```
ch2bgmedian
ch2volbg
ch2snr
repsd
ratio
directional
ln
ch1compensvol
ch2compensvol
compensatedratio
compensateddirectional
compensatedln
snrtotal
snrbygeoscore
replicas
```

Suppose that the previously created `marrayLayout` object is called 'Layout' and the `marrayInfo` object is called 'Gnames', then we can read in the raw data as follows:

```
> raw <- readMAGEraw(fname = "Iron.xml", datadir = datadir, name.Gf = "ch1vol",
+   name.Gb = "ch1bgmedian", name.Rf = "ch2vol", name.Rb = "ch2bgmedian",
+   layout = Layout, gnames = Gnames)
```

```
Reading 230MM-mod.txt
This file describes 1 hybridisation(s).
Reading 235MM-mod.txt
This file describes 1 hybridisation(s).
```

```
> print(raw)
```

```
Pre-normalization intensity data:          Object of class marrayRaw.
```

```
Number of arrays:          2 arrays.
```

```
A) Layout of spots on the array:
```

```
Array layout:          Object of class marrayLayout.
```

```
Total number of spots:          1152
Dimensions of grid matrix:          2 rows by 4 cols
Dimensions of spot matrices:          18 rows by 8 cols
```

```
Currently working with a subset of 1152 spots.
```

```
Control spots:
```

Notes on layout:

Array design is of type ARDES:A-EMBL-1

B) Samples hybridized to the array:

Object of class marrayInfo.

```
          maLabels          Cy3          Cy5
1 230MM-mod_Hyb_1 S:hela230hemin S:hela230desferal
2 235MM-mod_Hyb_1 S:caco2235hemin S:caco2235desferal
```

Number of labels: 2

Dimensions of maInfo matrix: 2 rows by 2 columns

Notes:

Description of the targets

C) Summary statistics for log-ratio distribution:

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
230MM-mod_Hyb_1	-4.18	-0.68	-0.45	-0.42	-0.19	10.35	44
235MM-mod_Hyb_1	-3.32	-0.10	0.20	0.16	0.46	3.85	45

D) Notes on intensity data:

E-MANP-1

The `marrayRaw` object is now stored in the object 'raw'. This function can again be called with a widget. Therefore you enter

```
> if (interactive()) {
+   widget.readMAGERaw()
+ }
```

Then a GUI appears as shown in Figure 2. In the first input box, you can choose the name of the raw and the targets object that will be created. In the second box you select the MAGE-ML file that contains the BioAssay, BioAssayData, and BioMaterial package. Again you can check if the necessary packages are available in the MAGE-ML file with the button 'Show Packages'.

The 'Foreground and Background Intensities' window gives you the possibility to enter the names of the columns containing the intensities. Here, you can also extract a list of available Quantitation types from the MAGE-ML file by hitting the 'Show Quantitation Type' button. The Quantitation types will then be shown in the R window.

In the following boxes you can enter the design objects (Layout and Gnames) of your microarrays. If everything is filled out properly, you just hit the 'Build' button and your `marrayRaw` object will be created.

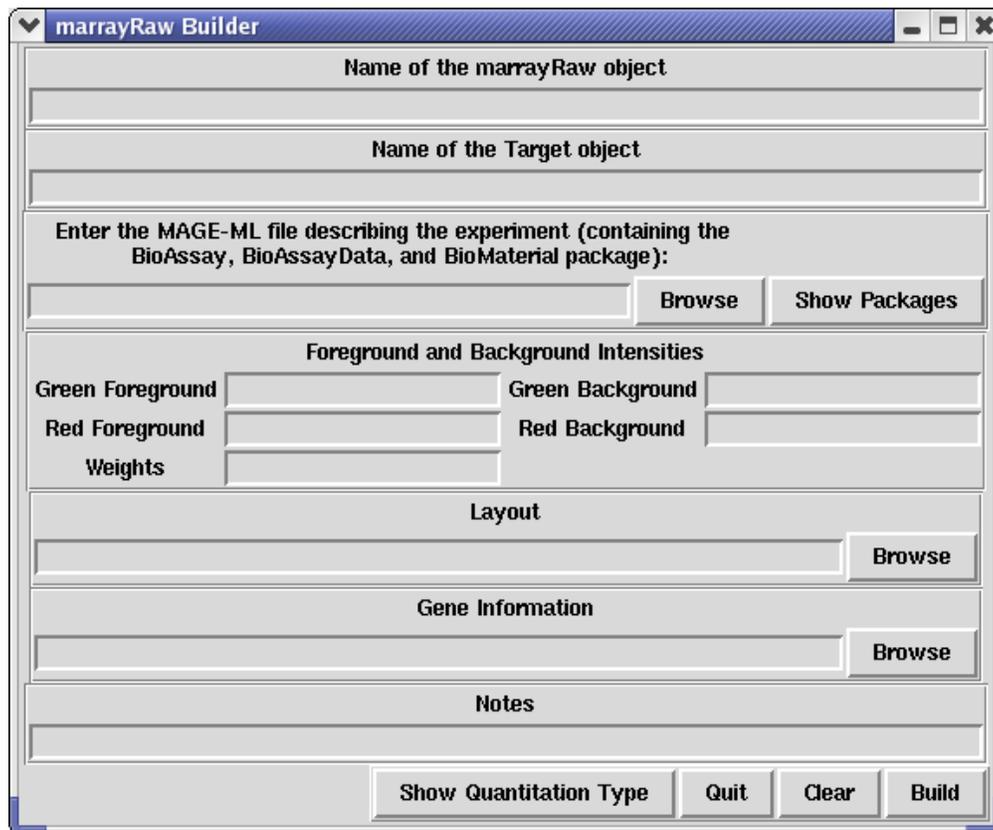


Figure 2: The widget appearing when calling `widget.readMageRaw()`.