

Tutorial for Bioconductor's `marrayClasses` package

Yee Hwa Yang¹ and Sandrine Dudoit²

August 25, 2004

1. Department of Medicine, University of California, San Francisco, jean@biostat.berkeley.edu
2. Division of Biostatistics, University of California, Berkeley.

Contents

1	Overview	1
2	Microarray classes	2
3	Creating and accessing slots of microarray objects	2
4	Basic microarray methods	3
4.1	Printing methods for microarray objects	3
4.2	Subsetting methods for microarray objects	6
4.3	Methods for accessing slots of microarray objects	9
4.4	Methods for assigning slots of microarray objects	10
4.5	Methods for coercing microarray objects	11
4.6	Functions for computing layout parameters	11

1 Overview

This document provides a short tutorial for the `marray` package, which is packages for diagnostic plots and normalization of cDNA microarray data. The `marray` packages contains basic class definitions and associated methods for pre- and post-normalization intensity data for batches of arrays. Greater details on the package are given in the `marrayClasses` document of the `inst/doc` directory.

Help files. As with any R package, detailed information on functions, classes and methods can be obtained in the help files. For instance, to view the help file for the class `marrayRaw` in a browser, use `help.start()` followed by `? marrayRaw` or alternately the dyadic class `? marrayRaw`.

Case study. We demonstrate the functionality of this collection of R packages using gene expression data from the Swirl zebrafish experiment. To load the Swirl dataset, use `data(swirl)`, and to view a description of the experiments and data, type `? swirl`.

```
> library(marray)
> data(swirl)
```

2 Microarray classes

The three main classes for cDNA microarray data are

marrayLayout: This class is used to keep track of important layout parameters for two-color cDNA microarrays. It contains slots for: the total number of spotted probe sequences on the array, the dimensions of the spot and grid matrices, the plate origin of the probes, information on spotted control sequences (e.g. probe sequences which should have equal abundance in the two target samples, such as housekeeping genes).

marrayRaw: This class represents pre-normalization intensity data for a batch of cDNA microarrays. A *batch* of arrays consists of a collection of arrays with the same layout ("marrayLayout"). The class contains slots for the green (Cy3) and red (Cy5) foreground and background intensities, the layout of the arrays, and descriptions of the target samples hybridized to the arrays and probe sequences spotted onto the arrays.

marrayNorm: This class represents post-normalization intensity data for a batch of cDNA microarrays. The class contains slots for the average log-intensities $A = \log_2 \sqrt{RG}$, the normalized log-ratios $M = \log_2 R/G$, the location and scale normalization values, the layout of the arrays, and descriptions of the target samples hybridized to the arrays and probe sequences spotted onto the arrays.

Other classes are **marrayInfo** which can be used to represent the Target or the Probes information. The function **slotNames** can be used to get information on the slots of a formally defined class or an instance of the class. For example, to get information of the slots for the **marrayLayout** class or on the slots for the object **swirl** use

```
> slotNames("marrayLayout")

[1] "maNgr"      "maNgc"      "maNsr"      "maNsc"      "maNspots"
[6] "maSub"      "maPlate"    "maControls" "maNotes"

> slotNames(swirl)

[1] "maRf"      "maGf"      "maRb"      "maGb"      "maW"      "maLayout"
[7] "maGnames"  "maTargets" "maNotes"
```

3 Creating and accessing slots of microarray objects

Creating new objects. The function **new** from the **methods** package may be used to create new objects from a given class. For example, to create an object of class **marrayInfo** describing the target samples in the Swirl experiment, one could use the following code

```
> zebra.RG <- as.data.frame(cbind(c("swirl", "WT", "swirl", "WT"),
+   c("WT", "swirl", "WT", "swirl")))
> dimnames(zebra.RG)[[2]] <- c("Cy3", "Cy5")
> zebra.samples <- new("marrayInfo", maLabels = paste("Swirl array ",
+   1:4, sep = ""), maInfo = zebra.RG, maNotes = "Description of targets for Swirl experiment")
> zebra.samples
```

```

An object of class "marrayInfo"
@maLabels
[1] "Swirl array 1" "Swirl array 2" "Swirl array 3" "Swirl array 4"

@maInfo
      Cy3   Cy5
1 swirl   WT
2   WT swirl
3 swirl   WT
4   WT swirl

@maNotes
[1] "Description of targets for Swirl experiment"

```

Slots which are not specified in `new` are initialized to the prototype for the corresponding class. These are usually "empty", e.g., `matrix(0,0,0)`. In most cases, microarray objects can be created automatically using the input functions and their corresponding widgets in the `marrayInput` package. These were used to create the object `swirl` of class `marrayRaw`.

functions in `marrayInput` were used to generate microarray objects `swirl.gnames`, `swirl.samples`, `swirl.layout`, and `swirl` for the Swirl experiment (the first two are of class `marrayInfo`, and the last two of class `marrayLayout` and `marrayRaw`, respectively).

Accessing slots. Different components or slots of the microarray objects may be accessed using the operator `@`, or alternately, the function `slot`, which evaluates the slot name. For example, to access the `maLayout` slot in the object `swirl` and the `maNgr` slot in the layout object `L`

```

> L <- slot(swirl, "maLayout")
> L@maNgr

[1] 4

```

4 Basic microarray methods

The following basic methods were defined to facilitate manipulation of microarray data objects. To see all methods available for a particular class, e.g., `marrayLayout`, or just the print methods

```

> showMethods(classes = "marrayLayout")
> showMethods("print", classes = "marrayLayout")

```

4.1 Printing methods for microarray objects

Since there is usually no need to print out fluorescence intensities for thousands of genes, the `print` method was overloaded for microarray classes by simple report generators. For an overview of the available microarray printing methods, type `methods ? print`, or to see all print methods for the session

```

> showMethods("print")

```

Function "print":
<not a generic function>

For example, summary statistics for an object of class `marrayRaw`, such as `swirl`, can be obtained by `print(swirl)` or simply `swirl`.

```
> swirl
```

An object of class "marrayRaw"

@maRf

	swirl.1.spot	swirl.2.spot	swirl.3.spot	swirl.4.spot
[1,]	19538.470	16138.720	2895.1600	14054.5400
[2,]	23619.820	17247.670	2976.6230	20112.2600
[3,]	21579.950	17317.150	2735.6190	12945.8500
[4,]	8905.143	6794.381	318.9524	524.0476
[5,]	8676.095	6043.542	780.6667	304.6190

8443 more rows ...

@maGf

	swirl.1.spot	swirl.2.spot	swirl.3.spot	swirl.4.spot
[1,]	22028.260	19278.770	2727.5600	19930.6500
[2,]	25613.200	21438.960	2787.0330	25426.5800
[3,]	22652.390	20386.470	2419.8810	16225.9500
[4,]	8929.286	6677.619	383.2381	786.9048
[5,]	8746.476	6576.292	901.0000	468.0476

8443 more rows ...

@maRb

	swirl.1.spot	swirl.2.spot	swirl.3.spot	swirl.4.spot
[1,]	174	136	82	48
[2,]	174	133	82	48
[3,]	174	133	76	48
[4,]	163	105	61	48
[5,]	140	105	61	49

8443 more rows ...

@maGb

	[,1]	[,2]	[,3]	[,4]
[1,]	182	175	86	97
[2,]	171	183	86	85
[3,]	153	183	86	85
[4,]	153	142	71	87
[5,]	153	142	71	87

8443 more rows ...

@maW

```

<0 x 0 matrix>

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

@maNgc
[1] 4

@maNsr
[1] 22

@maNsc
[1] 24

@maNspots
[1] 8448

@maSub
[1] TRUE

@maPlate
factor(0)
Levels:

@maControls
[1] Control Control Control Control Control
Levels: Control N
8443 more elements ...

@maNotes
[1] "No Input File"

@maGnames
An object of class "marrayInfo"
@maLabels
[1] "geno1" "geno2" "geno3" "3XSSC" "3XSSC"
8443 more elements ...

@maInfo
  "ID" "Name"
1 control  geno1
2 control  geno2

```

```
3 control  geno3
4 control  3XSSC
5 control  3XSSC
8443 more rows ...
```

```
@maNotes
```

```
[1] "C:/GNU/R/rw1041/library/marrayInput/data/fish.gal"
```

```
@maTargets
```

```
An object of class "marrayInfo"
```

```
@maLabels
```

```
[1] "81" "82" "93" "94"
```

```
@maInfo
```

	# of slide	Names	experiment	Cy3	experiment	Cy5	date	comments
1	81	swirl.1.spot		swirl	wild type		2001/9/20	NA
2	82	swirl.2.spot	wild type		swirl		2001/9/20	NA
3	93	swirl.3.spot		swirl	wild type		2001/11/8	NA
4	94	swirl.4.spot	wild type		swirl		2001/11/8	NA

```
@maNotes
```

```
[1] "C:/GNU/R/rw1041/library/marrayInput/data/SwirlSample.txt"
```

```
@maNotes
```

```
[1] ""
```

4.2 Subsetting methods for microarray objects

In many instances, one is interested in accessing only a subset of arrays in a batch and/or spots in an array. Subsetting methods "[" were defined for this purpose. For an overview of the available microarray subsetting methods, type `methods ? "["` or to see all subsetting methods for the session `showMethods("[")`. When using the "[" operator, the first index refers to spots and the second to arrays in a batch. Thus, to access the first 100 probe sequences in the second and third arrays in the batch `swirl` use

```
> swirl[1:100, 2:3]
```

```
An object of class "marrayRaw"
```

```
@maRf
```

	swirl.2.spot	swirl.3.spot
[1,]	16138.720	2895.1600
[2,]	17247.670	2976.6230
[3,]	17317.150	2735.6190
[4,]	6794.381	318.9524

```
[5,]      6043.542      780.6667
95 more rows ...
```

```
@maGf
```

```
      swirl.2.spot swirl.3.spot
[1,]      19278.770      2727.5600
[2,]      21438.960      2787.0330
[3,]      20386.470      2419.8810
[4,]       6677.619       383.2381
[5,]       6576.292       901.0000
95 more rows ...
```

```
@maRb
```

```
      swirl.2.spot swirl.3.spot
[1,]           136           82
[2,]           133           82
[3,]           133           76
[4,]           105           61
[5,]           105           61
95 more rows ...
```

```
@maGb
```

```
      [,1] [,2]
[1,]   175   86
[2,]   183   86
[3,]   183   86
[4,]   142   71
[5,]   142   71
95 more rows ...
```

```
@maW
```

```
<0 x 0 matrix>
```

```
@maLayout
```

```
An object of class "marrayLayout"
```

```
@maNgr
```

```
[1] 4
```

```
@maNgc
```

```
[1] 4
```

```
@maNsr
```

```
[1] 22
```

```
@maNsc
```

```

[1] 24

@maNspots
[1] 8448

@maSub
[1] TRUE TRUE TRUE TRUE TRUE
8443 more elements ...

@maPlate
[1] 1 1 1 1 1
Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
95 more elements ...

@maControls
[1] Control Control Control Control Control
Levels: Control N
95 more elements ...

@maNotes
[1] "No Input File"

@maGnames
An object of class "marrayInfo"
@maLabels
[1] "geno1" "geno2" "geno3" "3XSSC" "3XSSC"
95 more elements ...

@maInfo
      "ID" "Name"
1 control  geno1
2 control  geno2
3 control  geno3
4 control  3XSSC
5 control  3XSSC
95 more rows ...

@maNotes
[1] "C:/GNU/R/rw1041/library/marrayInput/data/fish.gal"

@maTargets
An object of class "marrayInfo"
@maLabels

```



```
[1] "82" "93"
```

```
@maInfo
```

	# of slide	Names	experiment	Cy3	experiment	Cy5	date	comments
2	82	swirl.2.spot	wild	type	swirl	2001/9/20	NA	
3	93	swirl.3.spot	swirl		wild type	2001/11/8	NA	

```
@maNotes
```

```
[1] "C:/GNU/R/rw1041/library/marrayInput/data/SwirlSample.txt"
```

```
@maNotes
```

```
[1] ""
```

4.3 Methods for accessing slots of microarray objects

A number of simple methods were defined to access slots of the microarray classes. Using such methods is more general than using the `slot` function or `@` operator. In particular, if the class definitions are changed, any function which uses the `@` operator will need to be modified. When using a method to access the data in the slot, only that particular method needs to be modified. Thus, to access the layout information for the array batch `swirl` one may also use `maLayout(swirl)`.

In addition, various methods were defined to compute basic statistics from microarray object slots. For instance, for memory management reasons, objects of class `marrayLayout` do not store the spot coordinates of each probe. Rather, these can be obtained from the dimensions of the grid and spot matrices by applying methods: `maGridRow`, `maGridCol`, `maSpotRow`, and `maSpotCol` to objects of class `marrayLayout`. Print-tip-group coordinates are given by `maPrintTip`. Similar methods were also defined to operate directly on objects of class `marrayRaw`, `marrayNorm`, `marraySpots`, and `marrayTwo`. The commands below may be used to display the number of spots on the array, the dimensions of the grid matrix, and the print-tip-group coordinates.

```
> swirl.layout <- maLayout(swirl)
```

```
> maNspots(swirl)
```

```
[1] 8448
```

```
> maNspots(swirl.layout)
```

```
[1] 8448
```

```
> maNgr(swirl)
```

```
[1] 4
```

```
> maNgc(swirl.layout)
```

```
[1] 4
```

```
> maPrintTip(swirl[1:10, 3])
```

```
[1] 1 1 1 1 1 1 1 1 1 1
```

4.4 Methods for assigning slots of microarray objects

A number of methods were defined to replace slots of microarray objects, without explicitly using the `@` operator or `slot` function. These make use of the `setReplaceMethod` function from the R `methods` package. As with the accessor methods just described, the assignment methods are named after the slots. For example, to replace the `maNotes` slot of `swirl.layout`

```
> maNotes(swirl.layout)
```

```
[1] "No Input File"
```

```
> maNotes(swirl.layout) <- "New value"
```

```
> maNotes(swirl.layout)
```

```
[1] "New value"
```

To initialize slots of an empty `marrayLayout` object

```
> L <- new("marrayLayout")
```

```
> L
```

An object of class "marrayLayout"

```
@maNgr
```

```
numeric(0)
```

```
@maNgc
```

```
numeric(0)
```

```
@maNsr
```

```
numeric(0)
```

```
@maNsc
```

```
numeric(0)
```

```
@maNspots
```

```
numeric(0)
```

```
@maSub
```

```
[1] TRUE
```

```
@maPlate
```

```
factor(0)
```

```
Levels:
```

```
@maControls
```

```
factor(0)
```

```
Levels:
```

```
@maNotes  
character(0)
```

```
> maNgr(L) <- 4
```

Similar methods were defined to operate on objects of class `marrayInfo`, `marrayRaw` and `marrayNorm`.

4.5 Methods for coercing microarray objects

To facilitate navigation between different classes of microarray objects, we have defined methods for coercing microarray objects from one class into another. A list of such methods can be obtained by `methods ? coerce`. For example, to coerce an object of class `marrayRaw` into an object of class `marrayNorm`

```
> swirl.norm <- as(swirl, "marrayNorm")
```

It is also possible to convert objects of class `marrayRaw` or `marrayNorm` into objects of class `exprSet` (see definition in the `Biobase` and `convert` package)

4.6 Functions for computing layout parameters

In some cases, plate information is not stored in `marrayLayout` objects when the data are first read into R. We have defined a function `maCompPlate` which computes plate indices from the dimensions of the grid matrix and number of wells in a plate. For example, the Swirl arrays were printed from 384-well plates, but the plate IDs were not stored in the `fish.gal` file. To generate plate IDs (arbitrarily labeled by integers starting with 1) and store these in the `maPlate` slot of the `marrayLayout` object use

```
> maPlate(swirl) <- maCompPlate(swirl, n = 384)
```

Similar functions were defined to generate and manipulate spot coordinates: `maCompCoord`, `maCompInd`, `maCoord2Ind`, `maInd2Coord`. The function `maGeneTable` produces a table of spot coordinates and gene names for objects of class `marrayRaw`, `marrayNorm`.