

Introduction to the Bioconductor marray package : Classes structure component

Sandrine Dudoit¹ and Yee Hwa Yang²

April 25, 2006

1. Division of Biostatistics, University of California, Berkeley,
<http://www.stat.berkeley.edu/~sandrine> 2. Department of Medicine, University of
California, San Francisco, jean@biostat.berkeley.edu

Contents

1	Overview	1
2	Object-oriented programming	2
3	Microarray classes	2
3.1	marrayLayout class	3
3.2	marrayInfo class	4
3.3	marrayRaw class	4
3.4	marrayNorm class	5
3.5	Creating and accessing slots of microarray objects	6
3.6	Testing the validity of an object	7
4	Basic microarray methods	7
4.1	Printing methods for microarray objects	8
4.2	Subsetting methods for microarray objects	9
4.3	Methods for accessing slots of microarray objects	38
4.4	Methods for assigning slots of microarray objects	39
4.5	Methods for coercing microarray objects	40
4.6	Functions for computing layout parameters	40

1 Overview

This document provides a tutorial on the class structures used in the `marray` package. The `marray` packages contains basic class definitions and associated methods for pre- and post-normalization intensity data for batches of arrays. To load the `marray` package in your R session, type `library(marray)`. 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`. Furthermore, se 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`.
Getting started:

2 Object-oriented programming

Microarray experiments generate large and complex multivariate datasets, which contain textual information on probe sequences (e.g. gene names, annotation, layout parameters) and mRNA target samples (e.g. description of samples, protocols, hybridization and scanning conditions), in addition to the primary fluorescence intensity data. Efficient and coordinated access to these various types of data is an important aspect of computing with microarray data. To facilitate the management of microarray data at different stages of the analysis process, a collection of microarray specific data structures or *classes* were defined (see also the Bioconductor package **Biobase** for microarray classes and methods for normalized data). The packages rely on the class/method mechanism provided by John Chambers' R **methods** package, which allows object-oriented programming in R. Broadly speaking, *classes* reflect how we think of certain objects and what information these objects should contain. Classes are defined in terms of *slots* which contain the relevant data for the application at hand. *Methods* define how a particular function should behave depending on the class of its arguments and allow computations to be adapted to particular classes, that is, data types. For example, a microarray object should contain intensity data as well as information on the probe sequences spotted on the array and the target samples hybridized to it. Useful methods for microarray classes include specializations of printing, subsetting, and plotting functions for the types of data represented by these classes.

The use of classes and methods greatly reduces the complexity of handling microarray data, by automatically coordinating various sources of information associated with microarray experiments.

3 Microarray classes

The *raw data* from a microarray experiment are the image files produced by the scanner; these are typically pairs of 16-bit tagged image file format (TIFF) files, one for each fluorescent dye (images usually range in size from a few megabytes (MB) to 10 or 20 MB for high resolution scans). Image analysis is required to extract foreground and background fluorescence intensity measurements for each spotted DNA sequence.

Here, we begin our analysis of microarray data with the output files of image processing packages such as **GenePix** or **Spot**. In what follows, red and green background intensities are denoted by R_b and G_b , respectively, and red and green foreground intensities by R_f and G_f , respectively. Background-corrected red and green fluorescence intensities are denoted by R and G , and M denotes the corresponding base-2 log-ratio, $M = \log_2 R/G$.

3.1 marrayLayout class

The term *array layout* refers to the layout of DNA probe sequences on the array, as determined by the printing process. In general, probe sequences are spotted on a glass microscope slide using an arrayer which has an $ngr \times ngc$ print-head, that is, a regular array of ngr rows and ngc columns of print-tips or pins. The resulting microarrays are thus partitioned into an $ngr \times ngc$ *grid matrix*. The terms *grid*, *sector*, and *print-tip-group* are used interchangeably in the microarray literature. Each grid consists of an $nsr \times nsc$ *spot matrix* that was printed with a single print-tip. DNA probes are usually printed sequentially from a collection of 384-well plates (or 96-well plates), thus, in some sense, plates are proxies for time of printing. In addition, a number of control probe sequences may be spotted on the array for normalization or other calibration purposes. The term *array batch* is used to refer to a collection of arrays with the same layout. Keeping track of spot layout information is essential for quality assessment of fluorescent intensity data and for normalization purposes.

Important layout parameters are the dimensions of the spot and grid matrices, and, for each probe on the array, its grid matrix and spot matrix coordinates. In addition, it is useful to keep track of gene names, plate origin of the probes, and information on the spotted control sequences (e.g. probe sequences which should have equal abundance in the two target samples, such as housekeeping genes). The class `marrayLayout` was designed to keep track of these various layout parameters and contains the following slots (the classes of the slots are listed below the slot names)

```
> getClassDef("marrayLayout")
```

Slots:

Name:	maNgr	maNgc	maNsr	maNsc	maNspots	maSub
Class:	numeric	numeric	numeric	numeric	numeric	logical

Name:	maPlate	maControls	maNotes
Class:	factor	factor	character

Extends: "ShowLargeObject"

maNgr: Object of class "numeric", number of rows for the grid matrix.

maNgc: Object of class "numeric", number of columns for the grid matrix.

maNsr: Object of class "numeric", number of rows for the spot matrices.

maNsc: Object of class "numeric", number of columns for the spot matrices.

maNspots: Object of class "numeric", total number of spots on the array, equal to $maNgr \times maNgc \times maNsr \times maNsc$.

maSub: Object of class "logical", indicating which spots are currently being considered.

maPlate: Object of class "factor", recording the plate origin of the spotted probe sequences.

maControls: Object of class "factor", recording the control status of the spotted probe sequences.

maNotes: Object of class "character", any notes concerning the microarray layout, e.g., printing conditions.

In addition, a number of *methods* were defined to compute other important layout parameters, such as print-tip, grid matrix, and spot matrix coordinates: **maPrintTip**, **maGridRow**, **maGridCol**, **maSpotRow**, and **maSpotCol** (see Section 4). No slots were defined for these quantities for memory management reasons.

3.2 marrayInfo class

Information on the target mRNA samples co-hybridized to the arrays is stored in objects of class **marrayInfo**. Such objects may include the names of the arrays, the names of the Cy3 and Cy5 labeled samples, notes on the hybridization and scanning conditions, and other textual information. Descriptions of the spotted probe sequences (e.g. matrix of gene names, annotation, notes on printing conditions) are also stored in object of class **marrayInfo**. The **marrayInfo** class is not specific to the microarray context and has the following definition

```
> getClassDef("marrayInfo")
```

Slots:

Name:	maLabels	maInfo	maNotes
Class:	character	data.frame	character

Extends: "ShowLargeObject"

3.3 marrayRaw class

Pre-normalization intensity data for a batch of arrays are stored in objects of class **marrayRaw**, which contain slots for the matrices of Cy3 and Cy5 background and foreground intensities (**maGb**, **maRb**, **maGf**, **maRf**), spot quality weights (**maW**), layout parameters of the arrays (**marrayLayout**), description of the probes spotted onto the arrays (**maGnames**) and mRNA target samples hybridized to the arrays (**maTargets**).

```
> getClassDef("marrayRaw")
```

Slots:

Name:	maRf	maGf	maRb	maGb	maW
Class:	matrix	matrix	matrix	matrix	matrix

Name:	maLayout	maGnames	maTargets	maNotes
Class:	marrayLayout	marrayInfo	marrayInfo	character

Extends: "ShowLargeObject"

maRf: Object of class "matrix", red foreground intensities, rows correspond to spotted probe sequences, columns to arrays in the batch.

maGf: Object of class "matrix", green foreground intensities, rows correspond to spotted probe sequences, columns to arrays in the batch.

maRb: Object of class "matrix", red background intensities, rows correspond to spotted probe sequences, columns to arrays in the batch.

maGb: Object of class "matrix", green background intensities, rows correspond to spotted probe sequences, columns to arrays in the batch.

maW: Object of class "matrix", spot quality weights, rows correspond to spotted probe sequences, columns to arrays in the batch.

maLayout: Object of class "marrayLayout", layout parameters for cDNA microarrays.

maNames: Object of class "marrayInfo", description of spotted probe sequences.

maTargets: Object of class "marrayInfo", description of target samples hybridized to the arrays.

maNotes: Object of class "character", any notes concerning the microarray experiments, e.g. hybridization or scanning conditions.

3.4 marrayNorm class

Post-normalization intensity data are stored in similar objects of class **marrayNorm**. These objects store the normalized intensity log-ratios **maM**, the location and scale normalization values (**maMloc** and **maMscale**), and the average log-intensities (**maA**). In addition, the **marrayNorm** class has a slot for the function call used to normalize the data, **maNormCall**. For more details on the creation of normalized microarray objects, the reader is referred to the vignette for the **marrayNorm** package.

```
> getClassDef("marrayNorm")
```

Slots:

Name:	maA	maM	maMloc	maMscale	maW
Class:	matrix	matrix	matrix	matrix	matrix

Name:	maLayout	maNames	maTargets	maNotes	maNormCall
Class:	marrayLayout	marrayInfo	marrayInfo	character	call

Extends: "ShowLargeObject"

maA: Object of class "matrix", average log-intensities (base 2) A , rows correspond to spotted probe sequences, columns to arrays in the batch.

maM: Object of class "matrix", intensity log-ratios (base 2) M , rows correspond to spotted probe sequences, columns to arrays in the batch.

maMloc: Object of class "matrix", location normalization values, rows correspond to spotted probe sequences, columns to arrays in the batch.

maMscale: Object of class "matrix", scale normalization values, rows correspond to spotted probe sequences, columns to arrays in the batch.

maW: Object of class "matrix", spot quality weights, rows correspond to spotted probe sequences, columns to arrays in the batch.

maLayout: Object of class "marrayLayout", layout parameters for cDNA microarrays.

maNames: Object of class "marrayInfo", description of spotted probe sequences.

maTargets: Object of class "marrayInfo", description of target samples hybridized to the arrays.

maNotes: Object of class "character", any notes concerning the microarray experiments, e.g. hybridization or scanning conditions.

maNormCall: Object of class "call", function call for normalizing the batch of arrays.

Most microarray objects contain an **maNotes** slots which may be used to store any string of characters describing the experiments, for examples, notes on the printing, hybridization, or scanning conditions.

3.5 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`.

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
```

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 on 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"
```

3.6 Testing the validity of an object

The function `validObject` from the R package `methods` may be used to test the validity of an object with respect to its class definition. This function has two arguments: `object`, the object to be tested; and `test`. If `test` is `TRUE`, the function returns a vector of strings describing the problems, if any.

```
> validObject(maLayout(swirl), test = TRUE)
```

```
[1] TRUE
```

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("show", 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 ? summary`, or to see all summary methods for the session

```
> showMethods("summary")
```

```
Function "summary":  
  object = "ANY"  
  object = "marrayLayout"  
  object = "marrayInfo"  
  object = "marrayRaw"  
  object = "marrayNorm"
```

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

```
> summary(swirl)
```

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

```
Number of arrays:      4 arrays.
```

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

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

```
Total number of spots:      8448
```

```
Dimensions of grid matrix:      4 rows by 4 cols
```

```
Dimensions of spot matrices:      22 rows by 24 cols
```

```
Currently working with a subset of 8448spots.
```

```
Control spots:
```

```
There are 2 types of controls :
```

```
Control probes
```

```
768      7680
```

```
Notes on layout:
```

```
No Input File
```

```
B) Samples hybridized to the array:
```

```
Object of class marrayInfo.
```


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

	comments
1	NA
2	NA
3	NA
4	NA

Number of labels: 4

Dimensions of maInfo matrix: 4 rows by 6 columns

Notes:

C:/GNU/R/rw1041/library/marrayInput/data/SwirlSample.txt

C) Summary statistics for log-ratio distribution:

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
swirl.1.spot	-2.73	-0.79	-0.58	-0.48	-0.29	4.42
swirl.2.spot	-2.72	-0.15	0.03	0.03	0.21	2.35
swirl.3.spot	-2.29	-0.75	-0.46	-0.42	-0.12	2.65
swirl.4.spot	-3.21	-0.46	-0.26	-0.27	-0.06	2.90

D) Notes on intensity data:

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"

Slot "maRf":

	swirl.2.spot	swirl.3.spot
[1,]	16138.7200	2895.1600
[2,]	17247.6700	2976.6230
[3,]	17317.1500	2735.6190
[4,]	6794.3810	318.9524
[5,]	6043.5420	780.6667
[6,]	21093.5100	12210.6900

[7,]	11325.5700	10695.5000
[8,]	16539.6100	12782.4200
[9,]	13287.2000	8841.6600
[10,]	5374.7080	894.3333
[11,]	5003.7620	515.1739
[12,]	5496.0950	588.3810
[13,]	16850.3600	10578.2400
[14,]	14441.1900	6767.5580
[15,]	9396.6130	2346.6790
[16,]	5535.5710	1051.0000
[17,]	984.6667	133.9821
[18,]	363.8400	142.3659
[19,]	6184.0000	1456.1900
[20,]	5816.1360	1322.5450
[21,]	6683.3810	1237.2270
[22,]	6025.9520	336.3810
[23,]	249.6667	119.7500
[24,]	221.5758	117.5143
[25,]	839.5600	135.5769
[26,]	10444.2000	8452.8440
[27,]	7068.7650	3032.6880
[28,]	2220.9320	851.9157
[29,]	13875.6800	5830.3540
[30,]	781.5238	167.8293
[31,]	802.5301	277.3370
[32,]	1436.1370	646.7500
[33,]	3910.6900	4933.1110
[34,]	12905.6500	6197.6960
[35,]	5864.1720	5396.9480
[36,]	22630.7500	14455.2200
[37,]	817.7652	545.3434
[38,]	5452.3540	5421.1600
[39,]	1054.4760	182.4474
[40,]	10903.3300	11882.6900
[41,]	15596.1200	15082.5400
[42,]	16356.6100	11997.7900
[43,]	684.7783	418.8528
[44,]	1563.1400	1232.0100
[45,]	9222.3540	11276.8200
[46,]	9708.6670	8655.7960
[47,]	588.0952	343.4286
[48,]	19457.6400	11943.1000
[49,]	9658.0500	1708.6920
[50,]	1496.3810	1392.0000
[51,]	8572.7140	1611.8800

[52,]	10312.3200	1658.0970
[53,]	4651.2170	1007.5000
[54,]	327.1786	160.6250
[55,]	12139.0000	4031.8650
[56,]	3103.6960	668.0455
[57,]	5968.1250	1404.7360
[58,]	17958.7700	12866.6600
[59,]	1894.0950	249.6667
[60,]	5621.5910	1369.5000
[61,]	7509.2610	2317.0250
[62,]	1597.3810	7394.2790
[63,]	17557.2000	2067.3810
[64,]	1927.5910	711.6957
[65,]	6663.5000	1242.7780
[66,]	13084.2000	7794.6740
[67,]	3531.2170	172.8542
[68,]	7771.0600	4459.9540
[69,]	871.5714	330.1905
[70,]	1184.6190	634.6190
[71,]	13852.2400	10288.2200
[72,]	7127.7430	2717.3980
[73,]	4782.0830	979.8864
[74,]	1082.8890	170.0000
[75,]	1957.4090	388.5714
[76,]	4175.7730	899.8077
[77,]	2469.0950	209.3611
[78,]	21232.7100	11359.7000
[79,]	6218.3410	1377.4940
[80,]	6164.2580	1580.5610
[81,]	13873.3000	6497.3830
[82,]	10492.1400	4554.0890
[83,]	10158.3100	1694.9410
[84,]	1733.3330	305.4348
[85,]	3961.8110	1174.8370
[86,]	794.0000	341.1250
[87,]	3291.6820	381.6667
[88,]	16694.0000	7443.3810
[89,]	9587.0870	3755.0370
[90,]	6762.8400	2455.1670
[91,]	11006.6700	3734.7620
[92,]	3560.1600	1094.2330
[93,]	8434.6670	3489.4000
[94,]	13451.9500	4989.5860
[95,]	6410.3210	2228.8630
[96,]	20840.1600	8639.1330

[97,]	4644.7600	674.5238
[98,]	1462.4760	160.5185
[99,]	498.5000	244.3333
[100,]	615.4400	255.0476

Slot "maGf":

	swirl.2.spot	swirl.3.spot
[1,]	19278.7700	2727.5600
[2,]	21438.9600	2787.0330
[3,]	20386.4700	2419.8810
[4,]	6677.6190	383.2381
[5,]	6576.2920	901.0000
[6,]	23769.1000	23377.9700
[7,]	14280.5700	8839.4550
[8,]	21246.3100	10529.9300
[9,]	16792.9700	7426.0710
[10,]	5023.9580	1095.9050
[11,]	5136.9520	609.8696
[12,]	5857.5240	540.0952
[13,]	19186.3000	18701.8900
[14,]	16336.1600	12034.6600
[15,]	10861.8900	4735.3330
[16,]	6255.6190	1050.7730
[17,]	398.6923	125.8750
[18,]	401.2800	113.2195
[19,]	6190.4760	1634.5710
[20,]	5821.8180	1401.3180
[21,]	6519.1900	1353.0000
[22,]	5814.8100	391.0476
[23,]	323.1429	115.5278
[24,]	231.5455	99.4000
[25,]	801.9200	227.8077
[26,]	20153.0700	9256.0560
[27,]	7237.8040	3347.7920
[28,]	2582.3070	1323.5540
[29,]	18180.0600	8318.6670
[30,]	905.3333	236.0976
[31,]	916.1687	307.7283
[32,]	1983.5880	1097.8570
[33,]	7469.1380	2808.1780
[34,]	13037.0000	11780.8600
[35,]	7443.3230	6927.5000
[36,]	30251.7000	14544.6900
[37,]	996.9217	446.0808
[38,]	5651.6880	5400.2230
[39,]	1090.9520	191.6579

[40,]	13169.3000	9836.5160
[41,]	18169.2300	12243.4900
[42,]	17881.7600	9598.6040
[43,]	920.8227	469.2147
[44,]	1670.0000	1405.8420
[45,]	10613.0500	6251.4520
[46,]	11913.5000	7279.2240
[47,]	675.8571	292.6667
[48,]	21280.3300	9411.6890
[49,]	9289.7000	3339.1350
[50,]	1405.5710	621.3636
[51,]	9315.7710	4400.7600
[52,]	10768.9100	5569.3870
[53,]	5701.3910	1371.5830
[54,]	581.2500	154.8333
[55,]	15414.4300	9679.0140
[56,]	3838.5220	744.4545
[57,]	6317.6500	2151.6230
[58,]	17548.6200	18144.9300
[59,]	1045.9050	539.9048
[60,]	5512.4550	2184.3550
[61,]	8839.6670	6692.5700
[62,]	1817.3330	10838.2400
[63,]	15584.5500	3028.1900
[64,]	2056.3180	895.7391
[65,]	6543.0420	1743.4810
[66,]	15058.7300	9255.9070
[67,]	3363.5220	125.8958
[68,]	8058.6540	6607.1380
[69,]	845.3810	709.9524
[70,]	1120.7620	940.6190
[71,]	16116.2200	18166.0600
[72,]	7704.8380	5161.7230
[73,]	6126.3500	2295.3640
[74,]	986.4444	210.1905
[75,]	1976.3640	584.8571
[76,]	4541.2270	1726.4230
[77,]	2101.2860	187.4444
[78,]	30593.6700	18597.4400
[79,]	7640.1360	2818.8440
[80,]	6726.2260	2631.8780
[81,]	17953.3000	12975.0800
[82,]	14121.2500	12474.1300
[83,]	13422.4400	5581.2350
[84,]	1665.5240	467.7391

[85,]	4089.9250	2505.0820
[86,]	923.2857	296.9167
[87,]	3563.1360	559.6296
[88,]	13719.5500	10272.9800
[89,]	10427.7400	10020.6500
[90,]	6793.5400	4856.3520
[91,]	10935.3800	7954.9180
[92,]	2774.6400	1014.3000
[93,]	7140.6540	6329.7470
[94,]	13120.6900	8697.0570
[95,]	7107.1070	2975.8430
[96,]	23280.5600	18000.2800
[97,]	4827.4800	1323.9520
[98,]	1606.7140	143.1111
[99,]	610.8636	345.8571
[100,]	680.3600	336.5714

Slot "maRb":

	swirl.2.spot	swirl.3.spot
[1,]	136	82
[2,]	133	82
[3,]	133	76
[4,]	105	61
[5,]	105	61
[6,]	105	61
[7,]	106	61
[8,]	107	63
[9,]	120	63
[10,]	120	63
[11,]	120	62
[12,]	120	61
[13,]	141	61
[14,]	141	61
[15,]	141	51
[16,]	148	49
[17,]	148	50
[18,]	148	50
[19,]	117	52
[20,]	95	52
[21,]	81	52
[22,]	80	47
[23,]	96	44
[24,]	96	44
[25,]	136	76
[26,]	133	76
[27,]	133	76

[28,]	128	61
[29,]	123	61
[30,]	118	61
[31,]	139	61
[32,]	154	63
[33,]	154	63
[34,]	154	63
[35,]	144	61
[36,]	144	58
[37,]	142	52
[38,]	142	52
[39,]	141	52
[40,]	148	49
[41,]	148	50
[42,]	148	50
[43,]	144	50
[44,]	144	50
[45,]	79	50
[46,]	79	50
[47,]	96	48
[48,]	96	48
[49,]	153	61
[50,]	153	61
[51,]	140	61
[52,]	128	61
[53,]	123	48
[54,]	127	52
[55,]	139	58
[56,]	154	63
[57,]	154	63
[58,]	161	63
[59,]	161	58
[60,]	161	48
[61,]	149	52
[62,]	149	52
[63,]	149	52
[64,]	148	47
[65,]	168	50
[66,]	168	50
[67,]	168	50
[68,]	144	50
[69,]	102	50
[70,]	99	50
[71,]	97	48
[72,]	97	48

[73,]	153	60
[74,]	153	60
[75,]	140	51
[76,]	128	51
[77,]	116	48
[78,]	127	52
[79,]	142	58
[80,]	154	62
[81,]	172	62
[82,]	172	62
[83,]	172	51
[84,]	161	48
[85,]	149	52
[86,]	149	52
[87,]	149	52
[88,]	185	50
[89,]	185	50
[90,]	185	50
[91,]	178	50
[92,]	144	50
[93,]	102	49
[94,]	99	50
[95,]	97	50
[96,]	101	50
[97,]	167	53
[98,]	167	53
[99,]	140	51
[100,]	137	51

Slot "maGb":

	[,1]	[,2]
[1,]	175	86
[2,]	183	86
[3,]	183	86
[4,]	142	71
[5,]	142	71
[6,]	142	71
[7,]	132	65
[8,]	132	69
[9,]	132	69
[10,]	132	69
[11,]	142	68
[12,]	142	60
[13,]	172	60
[14,]	172	59
[15,]	172	53

[16,]	162	53
[17,]	145	51
[18,]	145	50
[19,]	121	50
[20,]	108	57
[21,]	90	57
[22,]	93	57
[23,]	97	45
[24,]	97	45
[25,]	167	86
[26,]	187	86
[27,]	187	86
[28,]	163	71
[29,]	154	71
[30,]	151	71
[31,]	148	65
[32,]	166	69
[33,]	183	69
[34,]	183	69
[35,]	174	68
[36,]	174	60
[37,]	172	60
[38,]	172	59
[39,]	172	53
[40,]	183	53
[41,]	187	51
[42,]	187	50
[43,]	187	50
[44,]	163	57
[45,]	90	57
[46,]	93	57
[47,]	97	46
[48,]	97	47
[49,]	187	72
[50,]	187	72
[51,]	187	72
[52,]	164	66
[53,]	154	66
[54,]	157	66
[55,]	187	65
[56,]	192	59
[57,]	192	64
[58,]	186	64
[59,]	186	64
[60,]	174	56

[61,]	172	56
[62,]	172	56
[63,]	172	51
[64,]	183	51
[65,]	189	51
[66,]	189	51
[67,]	189	53
[68,]	163	57
[69,]	107	57
[70,]	102	57
[71,]	96	46
[72,]	108	47
[73,]	192	64
[74,]	192	63
[75,]	192	63
[76,]	175	61
[77,]	154	61
[78,]	164	61
[79,]	187	61
[80,]	192	54
[81,]	192	52
[82,]	186	52
[83,]	186	48
[84,]	174	52
[85,]	167	52
[86,]	167	52
[87,]	167	51
[88,]	183	51
[89,]	189	54
[90,]	189	54
[91,]	189	54
[92,]	163	54
[93,]	111	53
[94,]	102	48
[95,]	101	46
[96,]	108	47
[97,]	206	64
[98,]	206	60
[99,]	205	58
[100,]	175	58

Slot "maW":
<0 x 0 matrix>
Slot "maLayout":
An object of class "marrayLayout"
Slot "maSub":

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
Slot "maControls":
```

```

[91] probes probes probes probes probes probes probes probes probes
[100] probes
Levels: Control probes
Slot "maNgr":
[1] 4
Slot "maNgc":
[1] 4
Slot "maNsr":
[1] 22
Slot "maNsc":
[1] 24
Slot "maNspots":
[1] 8448
Slot "maNotes":
[1] "No Input File"
Slot "maGnames":
An object of class "marrayInfo"
Slot "maLabels":
  [1] "geno1"      "geno2"      "geno3"      "3XSSC"      "3XSSC"
  [6] "EST1"       "geno1"      "geno2"      "geno3"      "3XSSC"
 [11] "3XSSC"      "3XSSC"      "EST2"       "EST3"       "EST4"
 [16] "3XSSC"      "Actin"      "Actin"      "3XSSC"      "3XSSC"
 [21] "3XSSC"      "3XSSC"      "Actin"      "Actin"      "ath1"
 [26] "Cad-1"      "DeltaB"     "Dlx4"       "ephrinA4"   "FGF8"
 [31] "flk-1"      "GB3 Raldh2" "GB7 tbxa"   "groucho1"   "Her1"
 [36] "Her7 geno"  "HrT"        "MyoD"       "Ndr3"       "geno1"
 [41] "geno2"      "geno3"      "notch6"     "OR2"        "ORF2"
 [46] "geno1"      "geno2"      "geno3"      "1-A1"       "1-A5"
 [51] "1-A9"       "1-A13"      "1-A17"      "1-A21"      "1-E1"
 [56] "1-E5"       "1-E9"       "1-E13"      "1-E17"      "1-E21"
 [61] "1-I1"       "1-I5"       "1-I9"       "1-I13"      "1-I17"
 [66] "1-I21"      "1-M1"       "1-M5"       "1-M9"       "1-M13"
 [71] "1-M17"      "1-M21"      "2-A1"       "2-A5"       "2-A9"
 [76] "2-A13"      "2-A17"      "2-A21"      "2-E1"       "2-E5"
 [81] "2-E9"       "2-E13"      "2-E17"      "2-E21"      "2-I1"
 [86] "2-I5"       "2-I9"       "2-I13"      "2-I17"      "2-I21"
 [91] "2-M1"       "2-M5"       "2-M9"       "2-M13"      "2-M17"
 [96] "2-M21"      "3-A1"       "3-A5"       "3-A9"       "3-A13"
Slot "maInfo":
      "ID"      "Name"
1 control    geno1
2 control    geno2
3 control    geno3
4 control    3XSSC
5 control    3XSSC

```

6	control	EST1
7	control	geno1
8	control	geno2
9	control	geno3
10	control	3XSSC
11	control	3XSSC
12	control	3XSSC
13	control	EST2
14	control	EST3
15	control	EST4
16	control	3XSSC
17	control	Actin
18	control	Actin
19	control	3XSSC
20	control	3XSSC
21	control	3XSSC
22	control	3XSSC
23	control	Actin
24	control	Actin
25	control	ath1
26	control	Cad-1
27	control	DeltaB
28	control	Dlx4
29	control	ephrinA4
30	control	FGF8
31	control	flk-1
32	control	GB3 Raldh2
33	control	GB7 tbxa
34	control	groucho1
35	control	Her1
36	control	Her7 geno
37	control	HrT
38	control	MyoD
39	control	Ndr3
40	control	geno1
41	control	geno2
42	control	geno3
43	control	notch6
44	control	OR2
45	control	ORF2
46	control	geno1
47	control	geno2
48	control	geno3
49	fb16a01	1-A1
50	fb16a03	1-A5

51	fb16a05	1-A9
52	fb16a07	1-A13
53	fb16a09	1-A17
54	fb16a11	1-A21
55	fb17a01	1-E1
56	fb17a03	1-E5
57	fb17a05	1-E9
58	fb17a07	1-E13
59	fb17a09	1-E17
60	fb17a11	1-E21
61	fb18a01	1-I1
62	fb18a03	1-I5
63	fb18a05	1-I9
64	fb18a07	1-I13
65	fb18a09	1-I17
66	fb18a11	1-I21
67	fb19a01	1-M1
68	fb19a03	1-M5
69	fb19a05	1-M9
70	fb19a07	1-M13
71	fb19a09	1-M17
72	fb19a11	1-M21
73	fb20a01	2-A1
74	fb20a03	2-A5
75	fb20a05	2-A9
76	fb20a07	2-A13
77	fb20a09	2-A17
78	fb20a11	2-A21
79	fb21a01	2-E1
80	fb21a03	2-E5
81	fb21a05	2-E9
82	fb21a07	2-E13
83	fb21a09	2-E17
84	fb21a11	2-E21
85	fb22a01	2-I1
86	fb22a03	2-I5
87	fb22a05	2-I9
88	fb22a07	2-I13
89	fb22a09	2-I17
90	fb22a11	2-I21
91	fb23a01	2-M1
92	fb23a03	2-M5
93	fb23a05	2-M9
94	fb23a07	2-M13
95	fb23a09	2-M17

```

96 fb23a11      2-M21
97 fb24a01      3-A1
98 fb24a03      3-A5
99 fb24a05      3-A9
100 fb24a07     3-A13
Slot "maNotes":
[1] "C:/GNU/R/rw1041/library/marrayInput/data/fish.gal"
Slot "maTargets":
An object of class "marrayInfo"
Slot "maLabels":
[1] "82" "93"
Slot "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
Slot "maNotes":
[1] "C:/GNU/R/rw1041/library/marrayInput/data/SwirlSample.txt"
Slot "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` and `marrayNorm`. 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` package), see package `convert` package for more details.

```
> library(convert)
> as(normdata, "exprSet")
```

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` and `marrayNorm`.