Bioconductor's marrayTools package

Yee Hwa Yang

November 25, 2003

1. Division of Biostatistics, University of California, San Francisco,
`http://www.biostat.ucsf.edu/jean`

1 Overview

This document provides a brief guide to the *marrayTools* package, which is part of a suite of five packages for diagnostic plots and normalization of cDNA microarray data. Information on the other packages can be found in the vignettes of each package. There are three main components to this package. These are:

- to provide wrapper functions with many pre-set defaults to improve the ease of use. For example, the function `gpTools` provides a one line command for biologists to input gene expression data, perform basic diagnostic plots and calculate simple quality measures.
- to provide a simple framework for calculating basic univariate statistics such as mean, median and t-statistics for each gene across multiple samples.
- to provide a display of annotated information as a web–page. This is an extension of the `ll.htmlpage` function in the *annotate* package.

2 Wrapper functions: `gpTools` and `spotTools`

Installing the package. To install the *marrayTools* package for Windows operating systems, first start R and make sure you are connected to the internet. Next, select "**Packages**" from the menu and click on " **Install package(s) from Bioconductor...**". Lastly, select *marrayTools* from the pop–up windows and click click on "**OK**".

Getting started with `gpTools` and `spotTools`. Wrapper functions are provided to automatically read in data, generate a standard set of diagnostic plots and calculate a pre-defined set of quality information and perform `loess` normalization for specific types of image processing output files. The functions `gpTools` and `spotTools` are customized for image analysis files generated by the program *Spot* and *GenePix* respectively. Below is an outline to get you started:

1. Create a directory and move all the relevant image processing output files (e.g. `.gpr` files) and a file containing gene (or spot) descriptions (e.g. `.gal` file) to that directory.
2. Start R in that directory. Under the windows system, click on the menu **File** and select **Change dir...**. This will result in a pop–up window for you to enter the location of the

desired directory containing all the relevant image processing files. Alternatively, click on **browse** to nevigate and select the location of the directory. You can double check that you are in the correct directory by selecting **Display File(s)...** under the **File** menu.

3. To load the package in your R session, type:

```
> library(Biobase)

Welcome to Bioconductor
         Vignettes contain introductory material.  To view,
         simply type: openVignette()
         For details on reading vignettes, see
         the openVignette help page.

> library(marrayTools)

Loading required package: marrayNorm
Loading required package: marrayClasses
Loading required package: stepfun
Loading required package: marrayInput
Loading required package: annotate
Loading required package: genefilter
```

4. If your working directory contains *GenePix* files (`.gpr`), run the following command.

```
> data <- gpTools()
```

5. If your working directory contains *Spot* files (`.spot`), run the following command.

```
> data <- spotTools()
```

With the default arguments, these two functions will read in all the files residing in the directory, create a sub–directory containing the corresponding diagnostic plots (in JPEG format), create a sub–directory with basic quality information and four excel files. The four excel files are normA.xls, normM.xls, normMA.xls and quality.xls which has the normalized log–intensity (A) , log-ratios(M), both M and A values and quality information respectively for all image processing files.

This wrapper function automatically produces nine plots of

- pre– and post–normalization cDNA microarray data;
- MA–plots of pre– and post–normalization log–ratios M;
- color images of pre– and post–normalization log–ratios M;
- color images of average log–intensities A;
- histogram and overlay density of the signal to noise log–ratio for Cy5 and Cy3 channels; where the signal to noise ratios is defined as the foreground intensity (without background adjustment) over the background intensity; and

- dot–plots of M and A values for replicate controls probes.

In addition, this function automatically saves the figures to a file, in jpeg format. More detailed descriptions of all the arguments and options can be found in the help files. For instance, to view the help file for the function `gpTools` in a browser, use `help.start()` followed by `?gpTools`.

Here are some different ways of changing the arguments in `gpTools`.

- Perform diagnostic plots and normalizations with no background subtraction.

  ```
  > data <- gpTools(bg=FALSE)
  ```

- The microarray object class of 'marrayRaw' is returned rather than 'marrayNorm'. This also implies that no normalization was performed on the data.

  ```
  > data <- gpTools(raw=TRUE)
  ```

- Read in the data and perform normalization. No diagnostics plots or quality information are generated.

  ```
  > data <- gpTools(plot=FALSE, quality=FALSE)
  ```

Please contact us if you have other image processing output formats and would like a similar wrapper functions.

3 Basic statistics: `maStats`

The second component of this package provides a framework for calculating some basic univariate statistics for a simple microarray experiment. We will demonstrate the functionality of this collection qof R fucntions using gene expression data from the Swirl zebrafish experiment. The swirl data are included as part of the *marrayInput* package. To load the swirl dataset, use `data(swirl)`, and to view a description of the experiments and data, type `?swirl`. This data set contains only a `marrayRaw` object `swirl`.

```
> data(swirl)
> class(swirl)

[1] "marrayRaw"
```

There are two options to calculate the basic statistics. The first option follows closely concepts in the `genefilter` package. There are three steps that must be performed.

1. Create the estimation function(s).
2. Assemble them into a estimation function.
3. Apply the estimation function to the expression matrix.

For example, the following command calculates the mean and t-statistics for the expression data swirl.

```
> dataest <- maStat(swirl, funNames = list(mean = meanFun(), t = ttestFun()))
> dataest[1:10, ]

             mean  statistic    estimate parameter    p.value
 [1,] -0.21014067 -1.7204363 -0.21014067         3 0.18383896
 [2,] -0.16688524 -1.6437314 -0.16688524         3 0.19877744
 [3,] -0.11012624 -0.9803369 -0.11012624         3 0.39921304
 [4,] -0.20091733 -1.4679319 -0.20091733         3 0.23842394
 [5,] -0.22677420 -1.8420985 -0.22677420         3 0.16269781
 [6,] -0.49492507 -3.0486153 -0.49492507         3 0.05548858
 [7,] -0.16253512 -1.0440811 -0.16253512         3 0.37317351
 [8,] -0.13606291 -0.8910557 -0.13606291         3 0.43854478
 [9,] -0.16084311 -1.0442259 -0.16084311         3 0.37311626
[10,] -0.08094179 -0.4535045 -0.08094179         3 0.68098036
```

Alternatively you can place the name of the functions as an argument.

```
> dataest <- maStat(swirl, funNames = c("meanFun", "ttestFun"))

[1] "meanFun"  "ttestFun"

> dataest[1:10, ]

          meanFun  statistic    estimate parameter    p.value
 [1,] -0.21014067 -1.7204363 -0.21014067         3 0.18383896
 [2,] -0.16688524 -1.6437314 -0.16688524         3 0.19877744
 [3,] -0.11012624 -0.9803369 -0.11012624         3 0.39921304
 [4,] -0.20091733 -1.4679319 -0.20091733         3 0.23842394
 [5,] -0.22677420 -1.8420985 -0.22677420         3 0.16269781
 [6,] -0.49492507 -3.0486153 -0.49492507         3 0.05548858
 [7,] -0.16253512 -1.0440811 -0.16253512         3 0.37317351
 [8,] -0.13606291 -0.8910557 -0.13606291         3 0.43854478
 [9,] -0.16084311 -1.0442259 -0.16084311         3 0.37311626
[10,] -0.08094179 -0.4535045 -0.08094179         3 0.68098036
```

A small widget (Figure 1) is provided to allow users to specify the argument `funNames`. By default, the result is store under an object named `statres`. This can be changed with the argument `outputName`.

```
## Results save under the object name ‘‘statres’’
> widget.Stat(swirl)

## Results save under the object name ‘‘results’’
> widget.Stat(swirl, outputName="results")
```

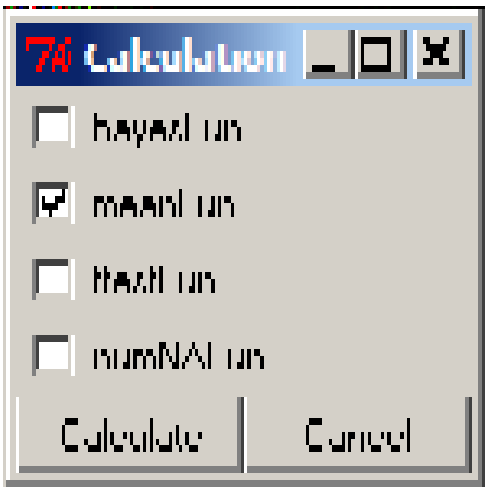

Figure 1: Screenshot of the widget for calculates various statistcs for genes in the array.

4 Web display: `table2html`

This section provides a brief description of how to display your table of genes and its corresponding annotated information as a html (web–page) file. This is an extension of the function `ll.htmlpage` provided in the *annotate* package. The basic function `table2html` displays a `data.frame` object as a html file. This web page has a few elements per genes that contains hyperlinks to various external databases. For the commonly used Affymetrix chips, you will be able to find annotation data from the various data packages at `http://www.bioconductor.org/data/data.html`. For less common Affymetrix chips, long-oligo or cDNA slides, you could build your own package using the package *AnnBuilder*, query from on-line data repository such as `SOURCE` or obtain the information from manufacturers of your cDNA or oligos probes.

The two main arguments for the function `table2html` is

- `restable`. This is a `data.frame` object that contains only the information you wish to display in the html file. Here, each row represents a different DNA spot and each column represents a different annotated infomration or other desired values to be include in the table.
- `mapURL`. This is a `matrix` object that provides a mapping between the column names in the data frame object and the various external databases.

The code below provides two mapping examples commonly used at the Sandler Functional Genomics Core at University of California, San Francisco (SFGL) and the Functional Genomics Core at University of Californa, Berkeley (UCBFGL).

```
> SFGL
```

```
          [,1]         [,2]
Name      "Name"       "none"
ID        "ID"         "operonST"
ACC       "ACC"        "SMDacc"
LocusLink "LocusLink"  "locuslink"
Cluster   "Cluster"    "unigeneMm"
```

```
LOCUSLINK "LOCUSLINK" "locuslink"
GenBank   "GenBank"   "genbank"
Grid      "Grid"      "cood"
Spot      "Spot"      "cood"
Row       "Row"       "cood"
Column    "Column"    "cood"
Block     "Block"     "cood"
```

```
> UCBFGL
```

```
       [,1]     [,2]
Name   "Name"   "pubmed"
ID     "ID"     "riken"
ACC    "ACC"    "SMDacc"
Grid   "Grid"   "cood"
Spot   "Spot"   "cood"
Row    "Row"    "cood"
Column "Column" "cood"
Block  "Block"  "cood"
```

The first column represents the name of the argument and the second column represents the index to the external database. The default is the value "none" and in this case, no hyperlink will be added. For example, in the matrix `UCBFGL`, the values associated with the column names "ID" are link to the Riken expression database *READ*, `http://read.gsc.riken.go.jp/`. The URL that corresponds to the various external databases are provided as a list object. These can be found with the following command.

```
> URLstring
```

Users can generate their own map with the function `mapGeneInfo`. For example, the following command indicates that values under the column name `ID` will be linked to the Riken's READ database and values under the column name `ll` will be linked to the Locus link data repository at NCBI.

```
> mapGeneInfo(ID = "riken", ll = "locuslink")
```

```
       [,1]     [,2]
Name   "Name"   "pubmed"
ID     "ID"     "riken"
ACC    "ACC"    "SMDacc"
ll     "ll"     "locuslink"
Grid   "Grid"   "cood"
Spot   "Spot"   "cood"
Row    "Row"    "cood"
Column "Column" "cood"
Block  "Block"  "cood"
```

Figure 2 shows a small widget that is included to allow users to enter the mapping information interactively. If you are using this widget, you will need to enter an object which contain description of spotted probe sequences. This can be an object of class 'matrix', 'data.frame' or 'marrayInfo'

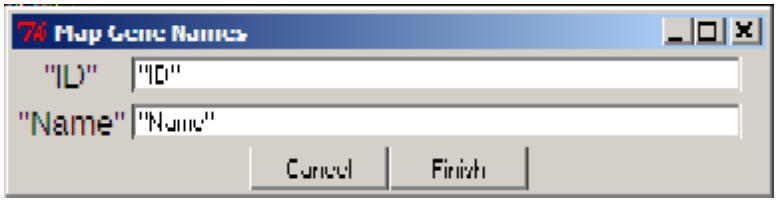

Figure 2: Screenshot of the widget which allow users to enter the mapping information between column names and external database.

```
> swirlmap <- mapGeneInfo(widget=TRUE, Gnames=maGeneTable(swirl))
```

In summary, the following sequence of commands illustrate the functions discussed above.

```
## Generating a data frame of desired infomation to be displayed as an
## html page.
> Info <- cbind(maGeneTable(swirl), maM(swirl))

## Generate the mapping matrix
> swirlmap <- mapGeneInfo(Name = "none", ID="genbank")

## Display the information as a web page.
> table2html(Info[100:110,], mapURL=swirlmap)
```

Use `?table2html` to find out more detail descriptions of all the aguments and options.

Note: Sweave. This document was generated using the `Sweave` function from the R *tools* package. The source file is in the `/inst/doc` directory of the package *marrayTools*.

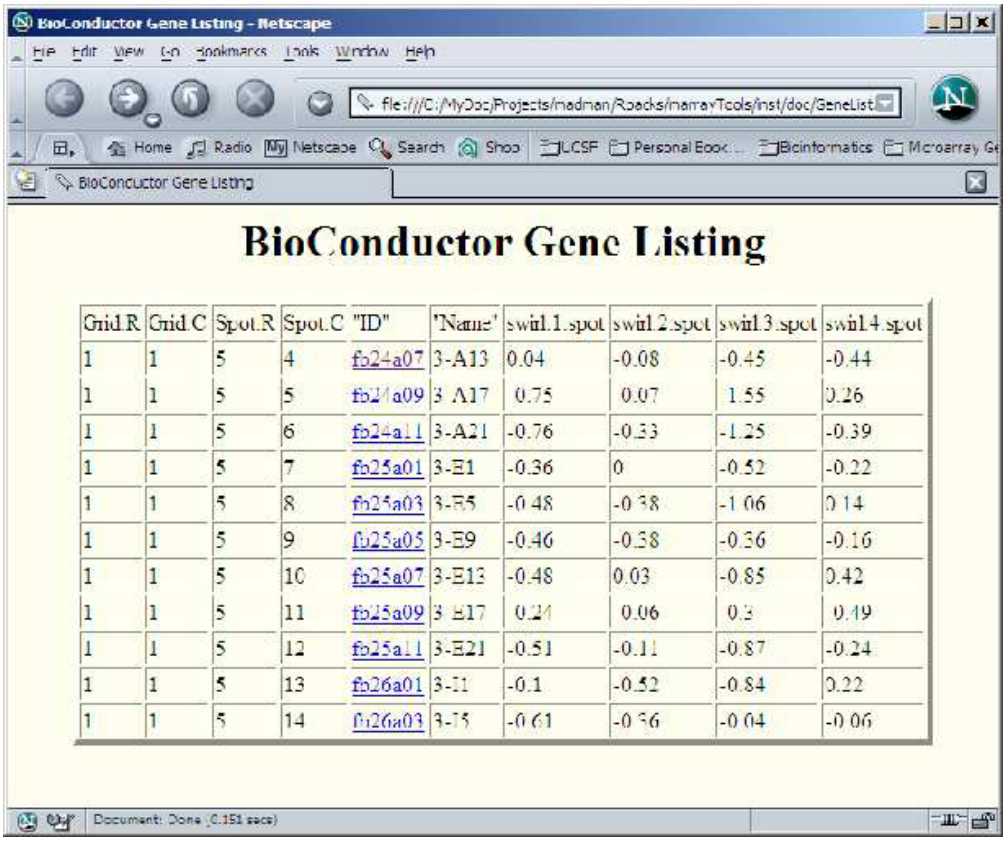

Figure 3: Screenshot of the resulting html page.