GS01 0163
Analysis of Microarray Data

Keith Baggerly and Kevin Coombes
Department of Bioinformatics & Computational Biology
UT M. D. Anderson Cancer Center
kabagg@mdanderson.org
kcoombes@mdanderson.org

15 November 2007

Lecture 23: R and Glass Microarrays

- Microarray Data Structures
- `marray` data structures
- `limma` data structures
- Toward a modular and efficient design
- Quantifying Glass Microarrays
- Getting down to business

Recall: Affymetrix analysis in BioConductor

- `exprSets` combine expression data and sample information
  - Can be linked in an efficient way to gene information
- `AffyBatch` objects hold the raw data
  - Easy to construct from a directory of CEL files
  - Gene annotations updated automatically
  - Useful quality control tools
- Structured, modular preprocessing with `expresso`
  - Background correction
  - Normalization
  - PM correction
  - Summarization

Glass arrays in BioConductor

BioConductor includes two different package bundles to deal with two-color glass microarrays: `marray` and `limma`.

Neither package uses the notion of an `exprSet`.

In both cases, the design seems to be less flexible and less modular than the tools for working with Affymetrix arrays.

`marray` data structures

The `marray` package uses four basic classes to hold the data from a collection of microarray experiments.

marrayInfo : holds sample information or gene information

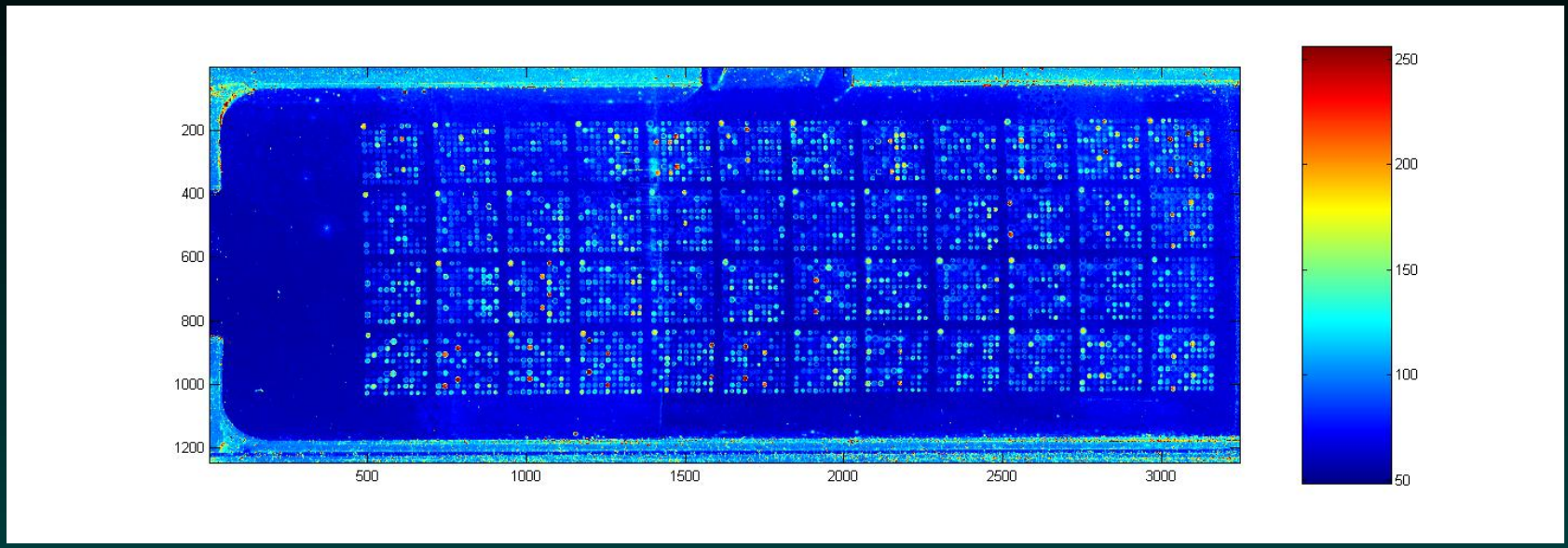
marrayLayout : describes the geometry of the array

marrayRaw : holds the raw array data

marrayNorm : holds array data after normalization

The primary processing function is `maNorm`, which allows you to try a limited number of normalization methods.

Geometry of glass microarray designs



As we have seen previously, glass microarrays are typically laid out in a hierarchical layout, containing a rectangle of grids, each of which is a rectangle of spots. Also, each grid is spotted on the array by a different physical pin.

`marrayLayout` slots

The `marray` package uses an `marrayLayout` object to describe the geometry using five numbers:

maNgr : number of grid rows

maNgc : number of grid columns

maNsr : number of spot rows

maNsc : number of spot columns

maNspots : number of spots

It is perhaps odd that they store the number of spots, since it seems to me that it should always be easily computable in terms of the other four parameters.

`marrayLayout` slots

The `marrayLayout` object may also include three additional vectors

maSub : a logical vector: are we currently interested in this spot?

maPlate : which plate did the robot get this spot from?

maControls : what kind of material is spotted here?

Metaphors appear to be mixed here: the `maPlate` and `maControls` vectors belong to the array design, and not to the specific analysis. The `maSub` object, however, seems to be an analysis-specific filter to let you focus on specific genes.

`marrayLayout` methods

They include methods to compute the following quantities, but they do not store them in the object:

maPrintTip : vector of print tips for the spots

maGridCol : vector of grid column locations

maGridRow : vector of grid row locations

maSpotCol : vector of spot column locations

maSpotRow : vector of spot row locations

`marrayRaw` slots

Raw expression data from glass microarrays is stored as an `marrayRaw` object, which contains:

- Four matrices of raw data (`maRf`, `maGf`, `maRb`, `maRb`) with red (R) and green (G) foreground (f) and background (b) estimates.
- An optional matrix (`maW`) of spot quality weights.
- `maLayout`, containing the array layout
- `maGnames`, containing the gene information
- `maTargets`, containing the sample information

`marrayRaw` methods

maA : vector of log intensities

maM : vector of log ratios

maLR : vector of background-corrected red log intensities

maLG : vector of background-corrected red log intensities

Note that there is no option to perform any form of background correction other than simply subtracting the values supplied by the image quantification software.

`marrayNorm` slots

Processed expression data from glass microarrays is stored as an `marrayNorm` object. These contain copies of the `maW`, `maLayout`, `maGnames`, and `maTargets` objects from the raw source data. In place of the raw measurements, these objects contain

maA : matrix of average log intensities

maM : matrix of log ratios

maMloc : localization normalization values

maMscale : scale normalization values

Normalization methods

In most cases, we want to normalize the data using `maNorm` (which is a wrapper around the more general function `maNormMain`). The basic function call looks like

```
> maNorm(my.raw.data, norm=method)
```

The normalization method must be specified as a character string, which must be one of the following: “none”, “median”, “loess”, “twoD”, “printTipLoess”, or “scalePrintTipMAD”. Unlike the approach taken with the Affymetrix arrays, there is no variable containing a list of normalization methods and no obvious way to add new methods. The more general method is extensible, but the way to extend it is poorly documented.

`limma` data structures

The `limma` package in BioConductor provides a different set of tools for glass microarrays.

RGList : raw microarray data as a list of arrays containing

- Four matrices, `R`, `G`, `Rb`, `Gb`, containing measurements.
- Optional components `weights`, `printer`, `genes`, `targets`.

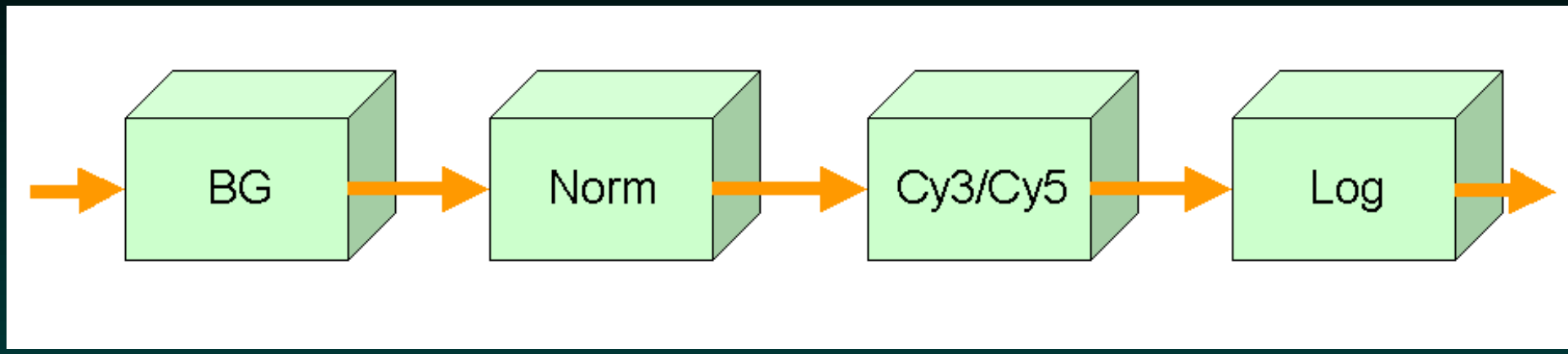
MAList : processed microarray data as a similar list with M and A components

`limma` normalization methods

The `limma` package has its own normalization routines (since they use different data structures than `marray`). Each has hard-coded option lists.

- normalizeBetweenArrays
- normalizeWithinArrays
- normalizeForPrintOrder
- normalizeRobustSpline
- normalizeMedians
- normalizeQuantiles

The processing pipeline



It should be possible to plug different algorithms in for each step in the pipeline.

It should be possible to add additional steps.

Ideally, it should be possible from the final object to reconstruct the processing history (which will be needed for the methods section of an article based on the analysis!).

Quantifying Glass Microarrays

So far, we have avoided describing how glass array data gets from the image quantification files into R and/or BioConductor.

The problem: There are lots of different software packages for image quantification. Unlike the Affymetrix world (where everything starts with the DAT and CEL files), this implies that there are lots of different formats that need to be understood by a general microarray analysis package.

In particular, when you construct an object to hold microarray data, you not only need to know the array design (i.e., the geometry and the gene annotations for each spot), but you need to know what software quantified the images.

Microarray Quantification Packages

There are a variety of programs available for quantifying arrays, including

- Free:
  - UCSF Spot
  - TIGR SpotFinder
- Commercial:
  - ArrayVision (Imaging Research, Inc.)
  - ImaGene (BioDiscovery, Inc.)
  - MicroVigene (Vigene Tech, Inc.)

Microarray Quantification Packages

Most manufacturers (e.g., Agilent or the Axon GenePix) of microarray scanners also supply quantification software.

- The critical issue to note is that every quantification package uses its own:
  - methods for finding, segmenting, and quantifying spots
  - scheme for labeling the spots
  - order for reporting the spots
  - names for the measurements it reports.

The only thing they have in common is that they are all able to export the data in tab-separated-values format, with rows representing spots and columns representing measurements (like location, foreground intensity, background intensity, etc.).

Quantifying Glass Microarrays

We are going to assume that we have somehow managed to get our hands on a set of quantification files from a batch of glass microarrays, and that we have determined what the individual columns mean. Our next goal is to figure out how to get this data into R and BioConductor so we can start doing something useful with it.

Reading data into `marray`

In `marray`, they handle this problem by using a variety of "read" functions:

- `read.GenePix`
- `read.Spot`
- `read.SMD`
- `read.marrayRaw`

Reading data into `limma`

In `limma`, there is a single "read" function

```
> read.maimages(files, source=SOMETHING)
```

This function uses hard-coded text strings to support different quantification packages; source can be one of

```
agilent     arrayvision    genepix
imagene     quantarray     smd
spot
```

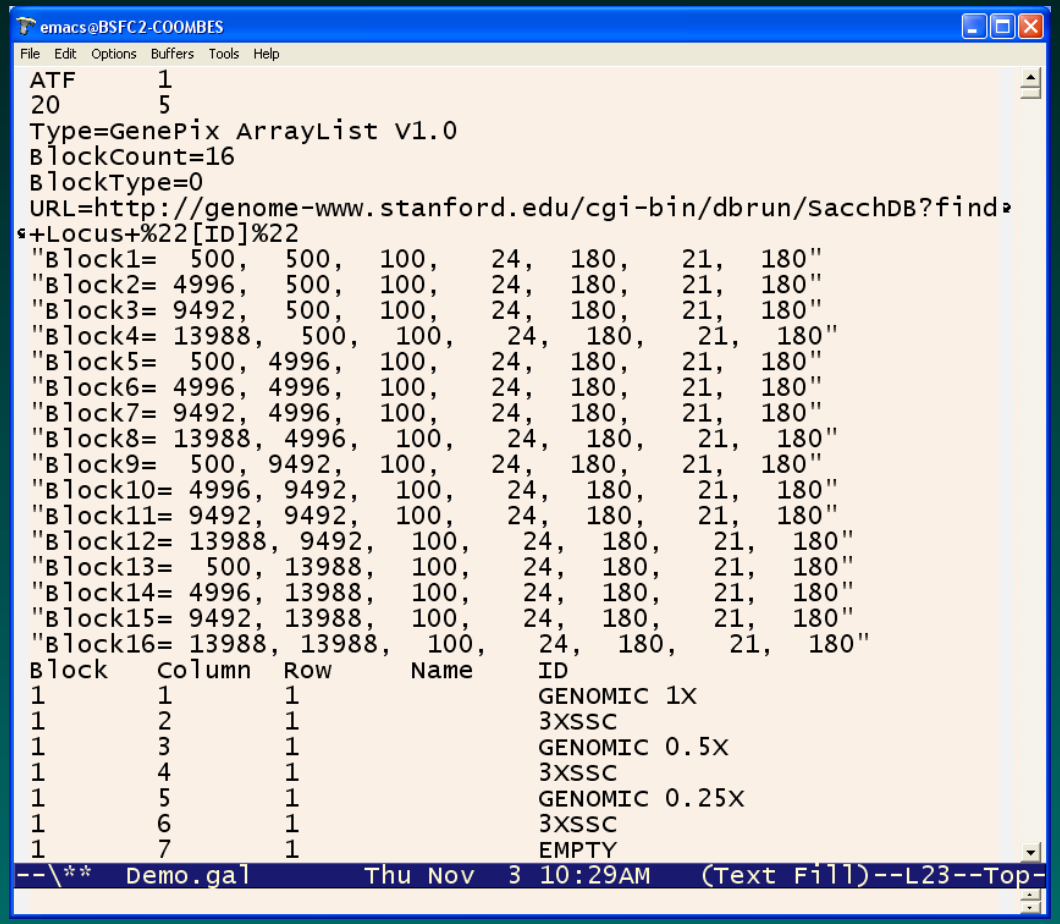
Getting down to business

An overview of the process:

1. Create an object that knows how to map spot label identifiers to gene information.
2. Create an object that understands the geometry of the array.
3. Create an object that records the sample information.
4. Load the raw data from all the arrays.
5. Process (background correct, normalize, summarize) the raw data.
6. Get to the fun part of the analysis. . . .

A sample GenePix GAL file

The Axon GenePix scanner software creates ".gal" files that describe the geometry of a glass microarray, along with the information that describes the gene probes at each spot.



The GenePix GAL file format

Axon describes the GAL file format on their web site:

http://www.moleculardevices.com/pages/software/gn_genepix_file_formats.html#gal

This is a special case of "Axon text format". The first line of the file (`ATF 1`) is required, and identifies the file format. The second line (`20 5`) is also required. It tells us, in this case, that there are 20 additional header lines before the main data starts, and that there are 5 columns of data. The third line (`Type=GenePix ArrayList V1.0`) is also required and identifies the type of GAL file format. Since they have only ever defined one version of the file format, this should be the same in all GAL files.

Block-heads

The next set of header lines is optional. In this case, they have chosen to tell us (`BlockCount=16`) that there are 16 blocks (or subgrids) contained on the array. The next line (`BlockType=0`) encodes the fact that these are rectangular blocks. The `URL=`$\cdots$ line gives an optional web site for more information.

Note that, even though the blocks=subgrids are themselves laid out in a rectangular pattern, the format at this point does not tell us what that pattern is. Axon numbers the blocks starting with number 1 in the upper left corner, marching across one row at a time before moving down.

Block descriptions

Next, each block is described by a line of the form

```
"Block1=  500,  500,  100,   24,  180,   21,  180
```

Each line contains 7 comma separated values describing the block. The first two entries give the X, Y position (in microns) of the top left corner of the block. The third value is the diameter of each spot in microns. The fourth value is the number of rows, and the fifth value is the spacing between spots in each row. The final two numbers are the number of columns and the spacing between spots in a column. Note that the geometry of the blocks can be inferred from the set of their X, Y positions.

Finally, the file contains a tab-separated set of information describing the spot locations and corresponding probe information.

Reading GAL files

The `marray` package includes a function that knows how to read GAL files, called, cleverly enough, `read.Galfile`. The simplest use is:

```
> demo.gal <- read.Galfile('demo.gal',
>       path='c://arrays/designs')
```

Warning: the following obvious attempt to read a GAL file somewhere other than the current directory will NOT work:

```
read.Galfile('c://arrays/designs/demo.gal')
```

Here the problem is that `read.Galfile` uses `path='.'` as the default value and always prepends the path to the file name.

Reading GAL files

After correctly reading the GAL file, the resulting object is a list:

```
> class(demo.gal)
[1] "list"
> attributes(demo.gal)
$names
[1] "gnames"   "layout"   "neworder"
> class(demo.gal$gnames)
[1] "marrayInfo"
> class(demo.gal$layout)
[1] "marrayLayout"
> class(demo.gal$neworder)
[1] "integer"
```

Reading GAL files

Since the GAL file contains the gene annotations (which have now been put into an `marrayInfo` object) and the geometry (put into an `marrayLayout` object), the function is able to extract both pieces of information. Thus, when working with array quantifications from Axon, you can accomplish the first two steps in a single function.

Other formats for gene information

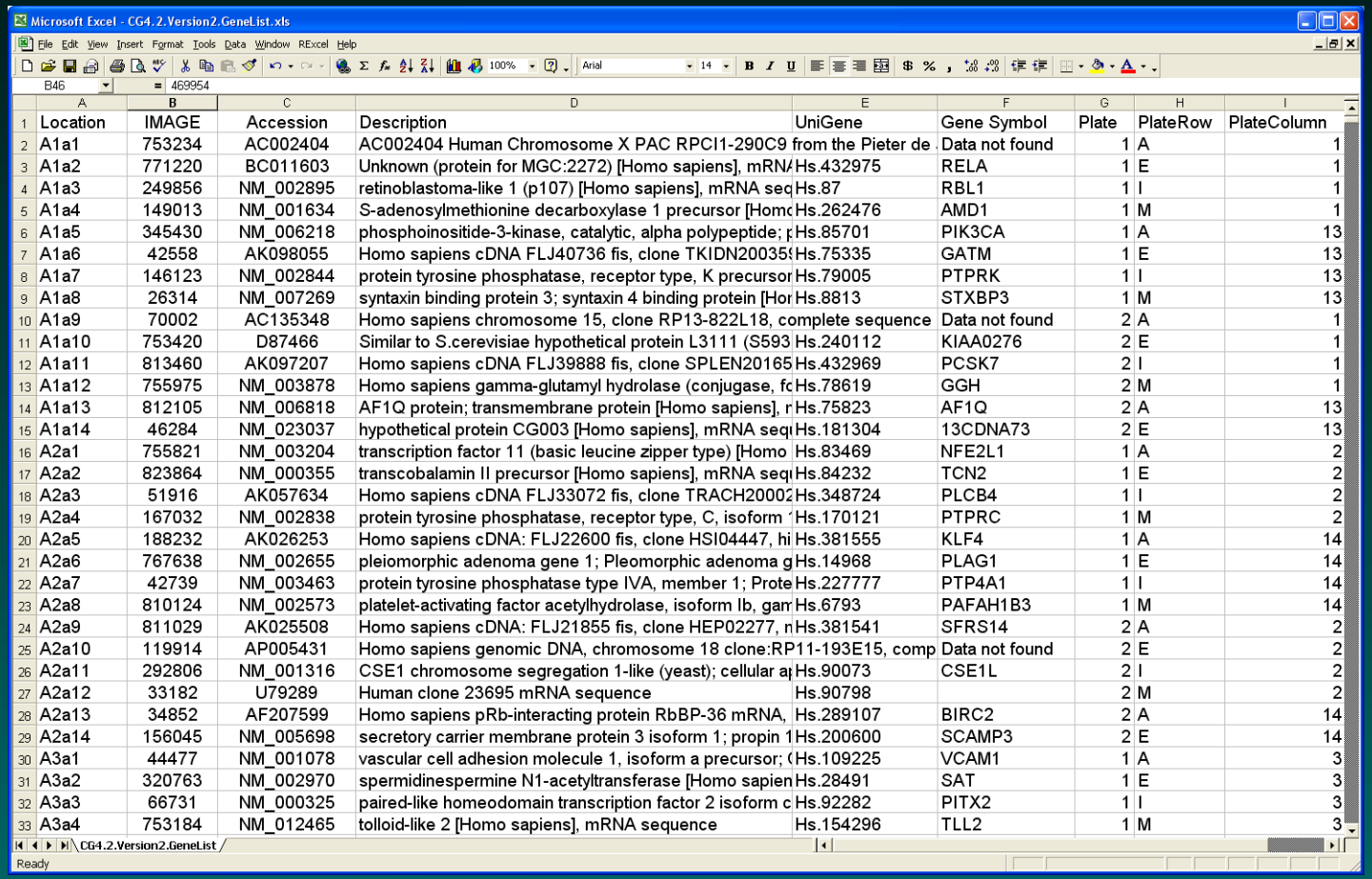
ArrayVision produces quantification files that include spot identifiers with absolutely no knowledge of the gene information:

Microsoft Excel - CG041372.0831.xls

	A	B	C	D	E	F	G	H
1		CG041372.0831.12A	CG041372.0831.12A	CG041372.0831.12A	CG041372.0831.12A	CG041372.0831.12A	CG041372.0831.12A	CG041372.083
2	Spot labels	VOL - Levels x mm2	SD - Levels	Pos X - mm	Pos Y - mm	Area - mm2	Bkgd	sVOL
3	A - 1 : A - 1	121.8033	933.81	4.133	20.025	0.045	78.383	
4	A - 1 : A - 2	79.532	165.88	4.573	20.065	0.045	76.454	
5	A - 1 : A - 3	86.1339	224.59	4.974	20.065	0.045	76.902	
6	A - 1 : A - 4	170.066	2000.56	5.334	19.985	0.045	77.89	9
7	A - 1 : A - 5	90.4956	347.89	5.754	20.065	0.045	79.505	1
8	A - 1 : A - 6	88.4942	298.6	6.175	20.065	0.045	77.934	
9	A - 1 : A - 7	83.7078	240.94	6.555	20.065	0.045	76.768	
10	A - 1 : A - 8	81.0398	175.38	6.975	20.065	0.045	78.069	
11	A - 1 : A - 9	86.0802	276.79	7.356	20.065	0.045	77.89	
12	A - 1 : A - 10	85.6227	287.72	7.756	20.065	0.045	77.62	
13	A - 2 : A - 1	92.6917	505.84	8.576	20.125	0.045	77.216	1
14	A - 2 : A - 2	77.8783	140.96	8.977	20.125	0.045	77.71	
15	A - 2 : A - 3	80.4141	165.48	9.377	20.125	0.045	77.755	
16	A - 2 : A - 4	83.2255	247.67	9.757	20.125	0.045	76.543	
17	A - 2 : A - 5	82.5701	209.27	10.158	20.125	0.045	76.05	
18	A - 2 : A - 6	83.5139	222.11	10.578	20.125	0.045	76.813	
19	A - 2 : A - 7	78.0505	158.77	10.958	20.125	0.045	77.172	
20	A - 2 : A - 8	84.9857	305.37	11.359	20.125	0.045	78.069	
21	A - 2 : A - 9	82.091	244.41	11.759	20.125	0.045	77.665	
22	A - 2 : A - 10	86.6474	306.51	12.159	20.125	0.045	76.185	1
23	A - 3 : A - 1	98.7383	590.54	13.1	20.125	0.045	77.665	2
24	A - 3 : A - 2	96.0727	1691.07	13.5	20.125	0.045	77.441	1
25	A - 3 : A - 3	79.157	170.83	13.9	20.125	0.045	78.069	
26	A - 3 : A - 4	87.6217	313.53	14.321	20.125	0.045	76.274	1
27	A - 3 : A - 5	78.0834	135.12	14.721	20.125	0.045	75.287	
28	A - 3 : A - 6	91.1013	445.77	15.121	20.125	0.045	76.005	1
29	A - 3 : A - 7	77.8703	145.79	15.522	20.125	0.045	77.71	
30	A - 3 : A - 8	80.6304	179.97	15.942	20.125	0.045	77.037	
31	A - 3 : A - 9	339.3012	5111.72	16.302	20.105	0.045	76.902	26
32	A - 3 : A - 10	89.9708	391.58	16.723	20.125	0.045	76.409	1
33	A - 4 : A - 1	108.134	706.71	17.523	20.085	0.045	77.531	3

Other formats for gene information

In this case, the core lab that produced the data also supplied a separate file with the gene annotations:



Microsoft Excel - CG4.2.Version2.GeneList.xls

File Edit View Insert Format Tools Data Window REXcel Help

B46 = 469954

	A	B	C	D	E	F	G	H	I
1	Location	IMAGE	Accession	Description	UniGene	Gene Symbol	Plate	PlateRow	PlateColumn
2	A1a1	753234	AC002404	AC002404 Human Chromosome X PAC RPCI1-290C9 from the Pieter de		Data not found	1	A	1
3	A1a2	771220	BC011603	Unknown (protein for MGC:2272) [Homo sapiens], mRN	Hs.432975	RELA	1	E	1
4	A1a3	249856	NM_002895	retinoblastoma-like 1 (p107) [Homo sapiens], mRNA seq	Hs.87	RBL1	1	I	1
5	A1a4	149013	NM_001634	S-adenosylmethionine decarboxylase 1 precursor [Hom	Hs.262476	AMD1	1	M	1
6	A1a5	345430	NM_006218	phosphoinositide-3-kinase, catalytic, alpha polypeptide;	Hs.85701	PIK3CA	1	A	13
7	A1a6	42558	AK098055	Homo sapiens cDNA FLJ40736 fis, clone TKIDN20035	Hs.75335	GATM	1	E	13
8	A1a7	146123	NM_002844	protein tyrosine phosphatase, receptor type, K precursor	Hs.79005	PTPRK	1	I	13
9	A1a8	26314	NM_007269	syntaxin binding protein 3; syntaxin 4 binding protein [Hor	Hs.8813	STXBP3	1	M	13
10	A1a9	70002	AC135348	Homo sapiens chromosome 15, clone RP13-822L18, complete sequence		Data not found	2	A	1
11	A1a10	753420	D87466	Similar to S.cerevisiae hypothetical protein L3111 (S593	Hs.240112	KIAA0276	2	E	1
12	A1a11	813460	AK097207	Homo sapiens cDNA FLJ39888 fis, clone SPLEN20165	Hs.432969	PCSK7	2	I	1
13	A1a12	755975	NM_003878	Homo sapiens gamma-glutamyl hydrolase (conjugase, fo	Hs.78619	GGH	2	M	1
14	A1a13	812105	NM_006818	AF1Q protein; transmembrane protein [Homo sapiens], r	Hs.75823	AF1Q	2	A	13
15	A1a14	46284	NM_023037	hypothetical protein CG003 [Homo sapiens], mRNA seq	Hs.181304	13CDNA73	2	E	13
16	A2a1	755821	NM_003204	transcription factor 11 (basic leucine zipper type) [Homo	Hs.83469	NFE2L1	1	A	2
17	A2a2	823864	NM_000355	transcobalamin II precursor [Homo sapiens], mRNA seq	Hs.84232	TCN2	1	E	2
18	A2a3	51916	AK057634	Homo sapiens cDNA FLJ33072 fis, clone TRACH20002	Hs.348724	PLCB4	1	I	2
19	A2a4	167032	NM_002838	protein tyrosine phosphatase, receptor type, C, isoform '	Hs.170121	PTPRC	1	M	2
20	A2a5	188232	AK026253	Homo sapiens cDNA: FLJ22600 fis, clone HSI04447, hi	Hs.381555	KLF4	1	A	14
21	A2a6	767638	NM_002655	pleiomorphic adenoma gene 1; Pleomorphic adenoma g	Hs.14968	PLAG1	1	E	14
22	A2a7	42739	NM_003463	protein tyrosine phosphatase type IVA, member 1; Prote	Hs.227777	PTP4A1	1	I	14
23	A2a8	810124	NM_002573	platelet-activating factor acetylhydrolase, isoform Ib, gam	Hs.6793	PAFAH1B3	1	M	14
24	A2a9	811029	AK025508	Homo sapiens cDNA: FLJ21855 fis, clone HEP02277, n	Hs.381541	SFRS14	2	A	2
25	A2a10	119914	AP005431	Homo sapiens genomic DNA, chromosome 18 clone:RP11-193E15, comp		Data not found	2	E	2
26	A2a11	292806	NM_001316	CSE1 chromosome segregation 1-like (yeast); cellular a	Hs.90073	CSE1L	2	I	2
27	A2a12	33182	U79289	Human clone 23695 mRNA sequence	Hs.90798		2	M	2
28	A2a13	34852	AF207599	Homo sapiens pRb-interacting protein RbBP-36 mRNA,	Hs.289107	BIRC2	2	A	14
29	A2a14	156045	NM_005698	secretory carrier membrane protein 3 isoform 1; propin 1	Hs.200600	SCAMP3	2	E	14
30	A3a1	44477	NM_001078	vascular cell adhesion molecule 1, isoform a precursor; (	Hs.109225	VCAM1	1	A	3
31	A3a2	320763	NM_002970	spermidinespermine N1-acetyltransferase [Homo sapien	Hs.28491	SAT	1	E	3
32	A3a3	66731	NM_000325	paired-like homeodomain transcription factor 2 isoform c	Hs.92282	PITX2	1	I	3
33	A3a4	753184	NM_012465	tolloid-like 2 [Homo sapiens], mRNA sequence	Hs.154296	TLL2	1	M	3

CG4.2.Version2.GeneList

Ready

Other formats for gene information

This example illustrates a more typical situation.

1. Neither of these text files explicitly describes the geometry of the array.

2. Neither file includes separate columns to identify the 'grid and subgrid row and column positions; these are embedded in the spot labels or locations.

3. The data file uses "Spot labels" of the form `A - 1 : A - 1`, while the annotations file describes the same "Location" in the form `A1a1`.

Reading the gene information

When we have a simple tab-separated file (like this one) containing the gene information, we can use it to produce a `marrayInfo` object.

```
> location <- 'C://arrays/designs'
> filename <- 'CG4.2.Version2.GeneList.txt'
> cg42 <- read.marrayInfo(file.path(location,
      filename), info.id=1:9, labels=6)
```

The `info.id` argument is optional; it is a list of the indices of the columns of the gene info file to include. The `labels` argument is also optional; it is the index of the column to use for labeling the gene. In our example, column 6 contains the gene symbols.

Checking the results

```
> cg42
An object of class "marrayInfo"
@maLabels
[1] ""       "RELA"   "RBL1"   "AMD1"   "PIK3CA"
10075 more elements ...

@maInfo
  Location  IMAGE Accession
1     A1a1 753234  AC002404
2     A1a2 771220  BC011603
3     A1a3 249856 NM_002895
4     A1a4 149013 NM_001634
5     A1a5 345430 NM_006218
```

```
    UniGene Gene Symbol Plate PlateRow PlateColum
1                              1        A         1
2 Hs.432975          RELA      1        E         1
3     Hs.87          RBL1      1        I         1
4 Hs.262476          AMD1      1        M         1
5  Hs.85701       PIK3CA      1        A        11
10075 more rows ...

@maNotes
[1] "C://arrays/designs/CG4.2.Version2.GeneList.t
```

Step 2: Getting the layout

Of course, we're still not done; we have to create an `marrayLayout` object with the geometry.

```
> temp <- as.character(cg42@maInfo$Location)
> temp <- temp[length(temp)]
> temp
[1] "D12o14"
> ngr <- which(LETTERS == substring(temp, 1, 1))
> ngc <- as.numeric(substring(temp, 2, 3))
> nsr <- which(letters == substring(temp, 4, 4))
> nsc <- as.numeric(substring(temp, 5, 6))
> cg42Layout <- new('marrayLayout',
+                   maNgr=ngr, maNgc=ngc,
+                   maNsr=nsr, maNsc=nsc,
+                   maPlate=factor(cg42@maInfo$Pl
```

Checking the layout

```
> summary(cg42Layout)
Array layout:  Object of class marrayLayout.

Total number of spots: 10080
Dimensions of grid matrix: 4 rows by 12 cols
Dimensions of spot matrices: 15 rows by 14 cols

Currently working with a subset of 10080spots.

Control spots:


Notes on layout:
```

How good are the gene annotations?

It is an unfortunate fact of life that the gene annotations for glass microarrays are rarely as good as the annotations for Affymetrix microarrays. The main difficulty is that we are dealing with many different manufacturers and software producers, so there is no central repository that has a vested interest in keeping the annotations up to date.

GAL files, for example, can contain varying degrees of information, varying highly in both the level of detail and the quality and accuracy of the annotations.

How good are the gene annotations?

As a general rule, you should try to get annotations that are as close as possible to describing the actual genetic material placed on the array. In particular, gene names, gene symbols, or UniGene cluster IDs are NOT primary identifiers of genomic material. You want something like:

- an IMAGE clone ID,
- a GenBank sequence identifier,
- or (in the case of long oligo arrays) the actual sequence spotted on the array.

A Two-Color Case Study

- Case Study Biology
- Getting Data
- Inferences from GPR Files
- Quality Checks
- Further Analysis

The Biology

Working with a case study. this follows Chapter 4 of Gentleman et al (2005), “Preprocessing Two-Color Spotted Arrays”, by Y.H. Yang and A.C. Paquet.

The dataset used here is a subset of a larger dataset described in Rodriguez et al (2004), “Differential gene expression by integrin β7+ and β7- memory T helper cells”, BMC Immunology, 5:13.

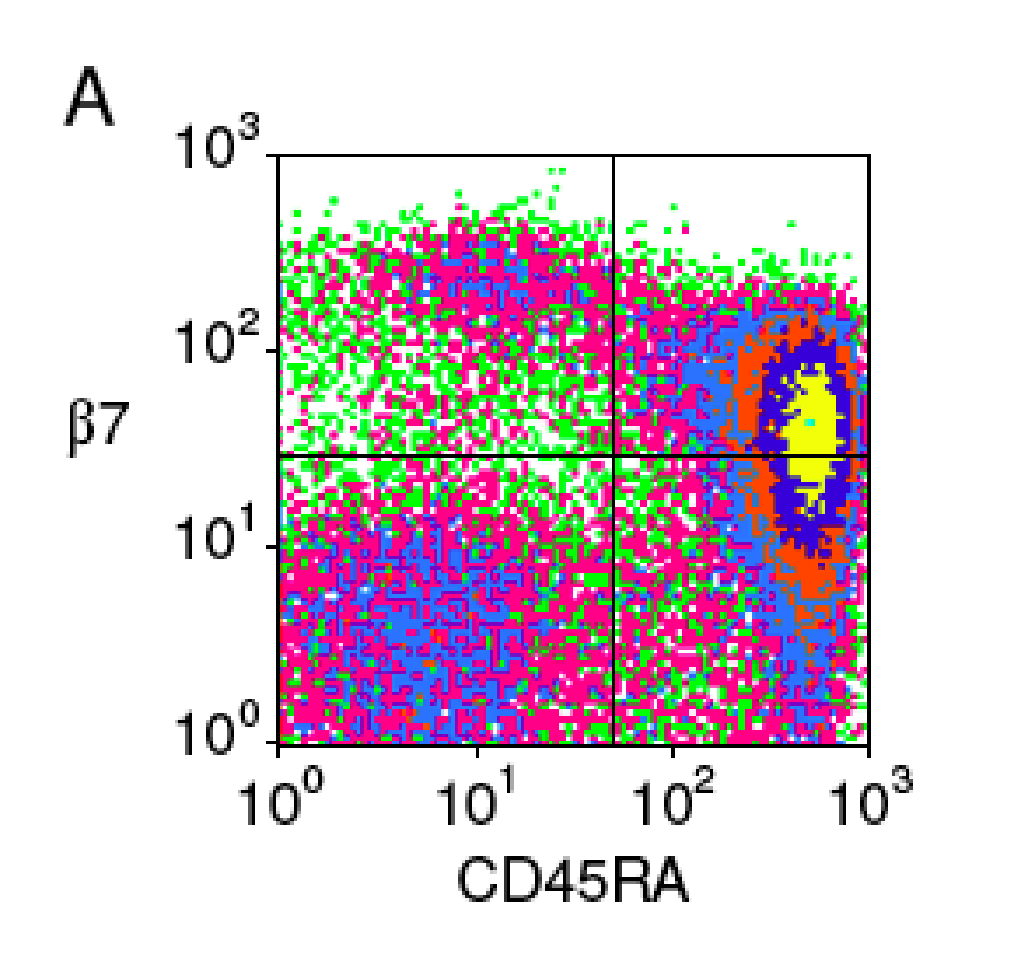
In that paper, they asked whether different types of helper cells were associated with the adhesion or migration of T cells.

How do we Get Cells?

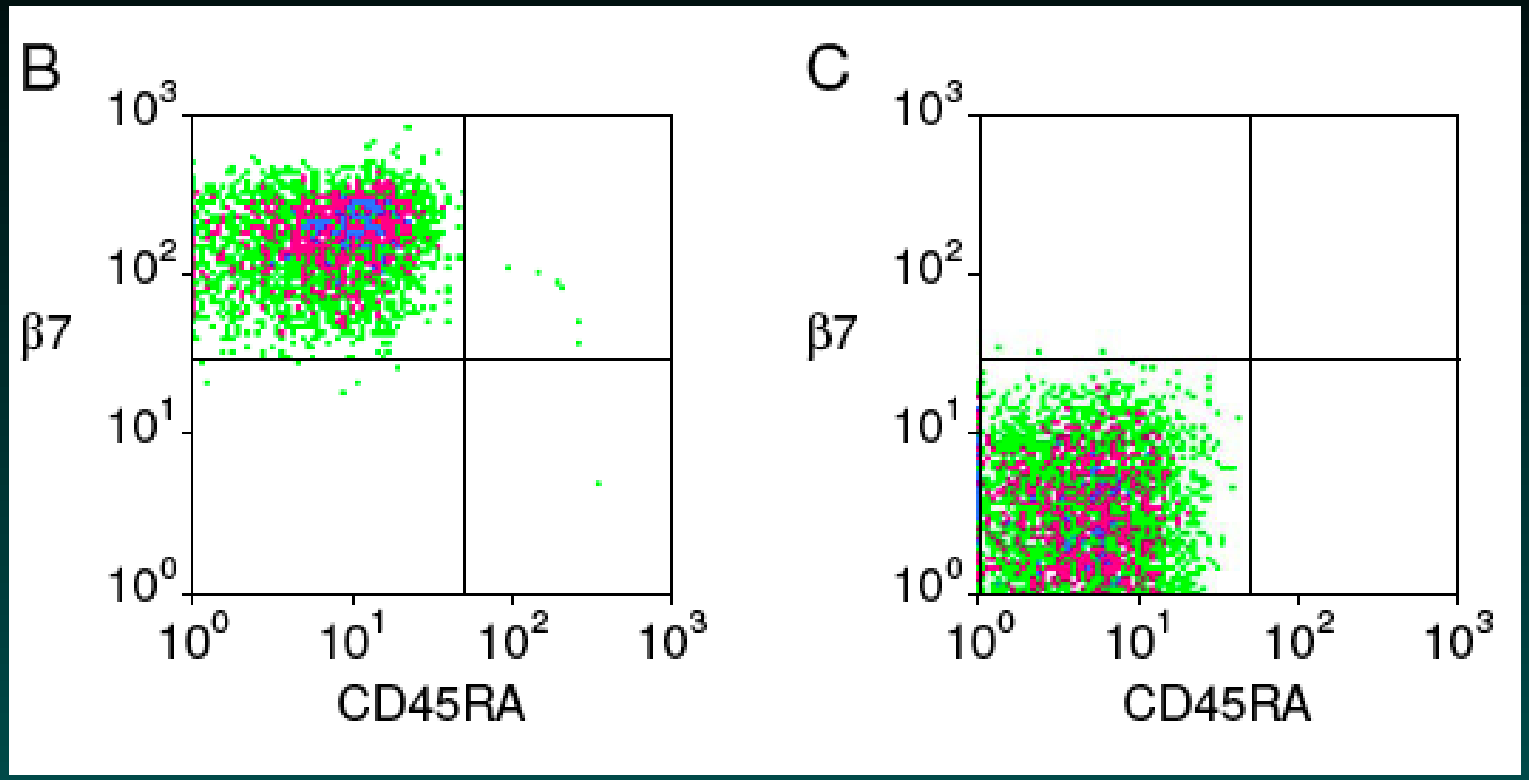
Extract CD4+ T cells, and derive enriched subpopulations that are $\beta 7$+ and $\beta 7$-. Cell subpopulations were obtained using flow cytometry.

Initially, cells are sorted by their levels of $\beta 7$ and CD45RA. High levels of CD45RA are not as interesting here, as their adhesion targets are already known. We want to focus on $\beta 7$ and see if we see separations there.

Cells Before Filtering



Cells After Filtering



After purification, the distributions are separated into our target groups.

Samples are Paired!

Extraction was done with samples from 9 individuals, so there is a natural data pairing.

Given the pairing, individual arrays were used to contrast the two by hybridizing β7+ in one channel and β7- in the other.

In all, 27 arrays were run, including at least 2 for each patient in a dye-swap arrangement.

The actual data is available from the Gene Expression Omnibus (GEO) maintained by the NCBI, with accession number GSE1039 (We'll come back to this).

Stuff Inferrable from GEO

sample, channel 1 (635nm), channel 2 (532nm), Patient ID, Gender (or is ch1 Cy3 and ch2 Cy5?)

GSM16665 - + 001 F GPL976 Hs_004_187_2
GSM16675 + - 001 F GPL976 Hs_004_186_2
GSM16679 - + 006 F GPL976 Hs_004_235
GSM16680 - + 009 F GPL976 Hs_004_189_1
GSM16681 + - 009 F GPL976 Hs_004_188
GSM16685 - + 001 F GPL978 6Hs.094
GSM16686 - + 001 F GPL978 6Hs.195.1 **
GSM16687 + - 003 F GPL978 6Hs.168 **

and so on. The ones with asterisks are contained in the subset we will look at today.

More on Methods

No data from patients 2 and 5.

The arrays used 70-mer oligos from Operon; there were 23184 spots on the arrays. Two different chip platforms were used when the experiment was run; these are available from GEO as

GPL976 UCSF 4Hs Human v.2 Oligo Array
GPL978 UCSF 6Hs Human v.2 Oligo Array

The RNA was subjected to 2 rounds of amplification using kits from Ambion.

All of the arrays were quantified using Axon's GenePix software, so we have gpr quantification files. The TIFF files are also available for download.

More on Methods, and our Subset

What other information would we like to have?
Run date? (scan date is available; this should be close)
Date of blood draw? (this is given in the TargetBeta7.txt file)
Gene information? (some of this is here)
Patient age? (this was there)

The data used here involves a subset of 6 arrays from this experiment. All 6 were of a single platform type, and had a common layout format.

Why were these 6 chosen?

Getting the Data

Let's get the 6 gpr files, and some TargetInfo and SpotInfo files

http://www.bioconductor.org/workshops/2005/BioC2005/labs/lab01/Data/integrinbeta7.zip

This zip file includes 6 gpr files, and a text file, TargetBeta7.txt, that contains sample information (eg, phenoData information). Eg:

```
FileNames          Subject ID #     Cy3       Cy5
6Hs.195.1.gpr      001              b7 -      b7 +
Hyb buffer         Hyb Temp (deg C) Hyb Time (h)
Ambion Hyb Slide           55       40
Date of Blood Draw         Amplification
2002.10.11                 R2 aRNA
```

Using R

The first step is simply to load a whole bunch of packages:

```
> library("marray");
> library("mclust");
> library("convert");
> library("arrayQuality");
> library("colorspace");
> library("grid");
> library("hexbin");
```

Getting the Sample Info

```
> TargetInfo <- read.marrayInfo("TargetBeta7.txt"
> TargetInfo
An object of class "marrayInfo"
@maLabels
[1] "6Hs.195.1.gpr" "6Hs.168.gpr"   "6Hs.166.gpr"
[5] "6Hs.194.gpr"   "6Hs.243.1.gpr"

@maInfo
      FileNames Subject ID #  Cy3  Cy5         Hyb
1 6Hs.195.1.gpr            1 b7 - b7 + Ambion Hyb
2   6Hs.168.gpr            3 b7 + b7 - Ambion Hyb
  Hyb Time (h) Date of Blood Draw Amplification
1           40          2002.10.11         R2 aRNA
2           40          2003.01.16         R2 aRNA
```

Getting the Numerical Info

Grab the data from the gpr files:

```
mraw <- read.GenePix(targets = TargetInfo);

# Note: this works on my PC. On my Mac laptop,
# I get the following error messages:

> mraw <- read.GenePix(targets = TargetInfo)
Error in if (skip > 0) readLines(file, skip) :
missing value where TRUE/FALSE needed
In addition: Warning messages:
1: input string 32 is invalid in this locale in:
  grep(pattern, x, ignore.case, extended, value,
2: input string 32 is invalid in this locale in:
  grep(pattern, x, ignore.case, extended, value,
```

What Can be Inferred?

So, what does our marrayRaw object contain at this point?

Let's take a look at the individual slots here.

```
> slotNames(mraw)
[1] "maRf"        "maGf"        "maRb"        "maGb"
[6] "maLayout"    "maGnames"    "maTargets"   "maNotes"
```

Of these, the first 5 are the basic quantification information, extracted from the gpr files. All of them are 23184 by 6 in size. The others are the associated layout and annotation files. Let's extract these and find out a bit more about them.

Summary, Part 1 – Layout

```
> summary(mraw)
Pre-normalization intensity data:
  Object of class marrayRaw.

Number of arrays:  6 arrays.

A) Layout of spots on the array:
Array layout:  Object of class marrayLayout.

Total number of spots: 23184
Dimensions of grid matrix: 12 rows by 4 cols
Dimensions of spot matrices: 23 rows by 21 cols

Currently working with a subset of 23184spots.
```

More Layout

```
Control spots:
There are   5 types of controls :

   Buffer    Empty Negative Positive   probes
        3     1328      225      204    21424
```

```
Notes on layout:
```

The layout can be inferred from the gpr files! This is not too suprising, as every row of a gpr file contains entries for grid row, grid col, spot row, and spot col. As a side note, what is the precise order?

Layout Ordering

```
> zedL <- mraw@maLayout
> zedLSC <- maSpotCol(zedL); zedLSR <- maSpotRow(
> zedLGR <- maGridRow(zedL); zedLGC <- maGridCol(
> zedLcoords <- cbind(zedLGR,zedLGC,zedLSR,zedLSC

> zedLcoords[1:25,]
      zedLGR zedLGC zedLSR zedLSC
 [1,]      1      1      1      1
 [2,]      1      1      1      2
 [3,]      1      1      1      3
...
[20,]      1      1      1     20
[21,]      1      1      1     21
[22,]      1      1      2      1
```

Summary Part 2 – Sample Info

```
B) Samples hybridized to the array:
Object of class marrayInfo.

        maLabels     FileNames SubjectID  Cy3  Cy5
1 6Hs.195.1.gpr 6Hs.195.1.gpr         1 b7 - b7 +
2   6Hs.168.gpr   6Hs.168.gpr         3 b7 + b7 -
..
  Date of Scan
1   2003.07.25
2   2003.08.07
..
```

Since we supplied the marrayInfo file in the call to read.GenePix, this is imported from there.

Summary Part 3 – Array Summaries

```
C) Summary statistics for log-ratio distribution:
                 Min. 1st Qu. Median  Mean 3rd Qu. M
6Hs.195.1.gpr -6.13   -1.00  -0.52 -0.50   -0.08 
6Hs.168.gpr   -7.08   -0.80  -0.21 -0.23    0.34 
6Hs.166.gpr   -7.07   -1.25  -0.64 -0.62   -0.02 
6Hs.187.1.gpr -9.81   -0.92  -0.60 -0.55   -0.25 
6Hs.194.gpr   -5.93    0.00   0.44  0.53    0.90 
6Hs.243.1.gpr -6.38   -1.13  -0.69 -0.64   -0.21 
```

Log ratios – what direction is the default? Cy3/Cy5? Cy5/Cy3? (the latter, according to documentation)

Summary Part 4 – Notes

D) Notes on intensity data:
GenePix Data

Ok, that dealt with most of the microarray structure itself.

What happens if we ask about the gene names? This is what we really want, so that we can understand the biology.

Annotation

```
> mraw@maGnames[1:2,]
An object of class "marrayInfo"
@maLabels
[1] "H200000297" "H200000303"
@maInfo
                      ID
H200000297 H200000297
                    Name
H200000297 OVGP1 - Oviductal glycoprotein 1, 120kD
@maNotes
[1] ""
```

again, these are read in from the gpr files. The first column here, the maLabels, is the Operon-supplied identifier for that specific oligo, and as such it should be unique.

Getting the Data: TMTOWTDI

Assembling an marrayRaw object need not be hard.

So, what if you're working with a Mac?

This marrayRaw object and a few other things are available as a package from BioConductor called "beta7". I had to run a search at the top level of BioConductor to find this; it is part of the "Data" page associated with the monograph. I downloaded the gzipped tar (.tar.gz) file and did an install from local source.

http://www.bioconductor.org/docs/mogr/data

library("beta7"); data(beta7);

loads an marrayRaw object (called beta7) with info on the 6 selected arrays.

How was Data Reported?

Table 1: Gene transcripts with higher expression in $\beta7^+$ versus $\beta7^-$ $CD4^+$ $CD45RA^-$ T helper cells*

Symbol	Name	Accession	Fold Difference	P value
CCR9	chemokine (C-C motif) receptor 9	NM_031200	+3.0	< 0.01
CCL5	chemokine (C-C motif) ligand 5	NM_002985	+2.4	< 0.01
RAM2	transcription factor RAM2	NM_018719	+2.2	< 0.01
LRRN3	leucine rich repeat neuronal 3	AL442092	+2.1	< 0.01
GFI1	growth factor independent 1	NM_005263	+1.8	< 0.01
ITGA4	integrin, alpha 4 (CD49D)	NM_000885	+1.7	< 0.01
CD1C	CD1C antigen, c polypeptide	NM_001765	+1.7	< 0.01
KLRB1	killer cell lectin-like receptor subfamily B, member 1	NM_002258	+1.7	< 0.01
LAIR1	leukocyte-associated Ig-like receptor 1	NM_002287	+1.7	< 0.01
RRM2	ribonucleotide reductase M2 polypeptide	NM_001034	+1.6	< 0.01
--	Homo sapiens cDNA FLJ32290 fis, clone PROST2000463	AK056852	+1.6	< 0.01
HHL	expressed in hematopoietic cells, heart, liver	NM_014857	+1.6	0.02
IL18RAP	interleukin 18 receptor accessory protein	NM_003853	+1.6	< 0.01
SREBF1	sterol regulatory element binding transcription factor 1	NM_004176	+1.6	< 0.01
KLRG1	killer cell lectin-like receptor subfamily G, member 1	NM_005810	+1.5	< 0.01
LGALS2	lectin, galactoside-binding, soluble, 2 (galectin 2)	NM_006498	+1.5	0.01

* Includes all transcripts with fold difference $\geq +1.5$ and adjusted $P < 0.05$. Positive fold difference values indicate higher expression on $\beta7^+$ cells.

There are some unique identifiers here!

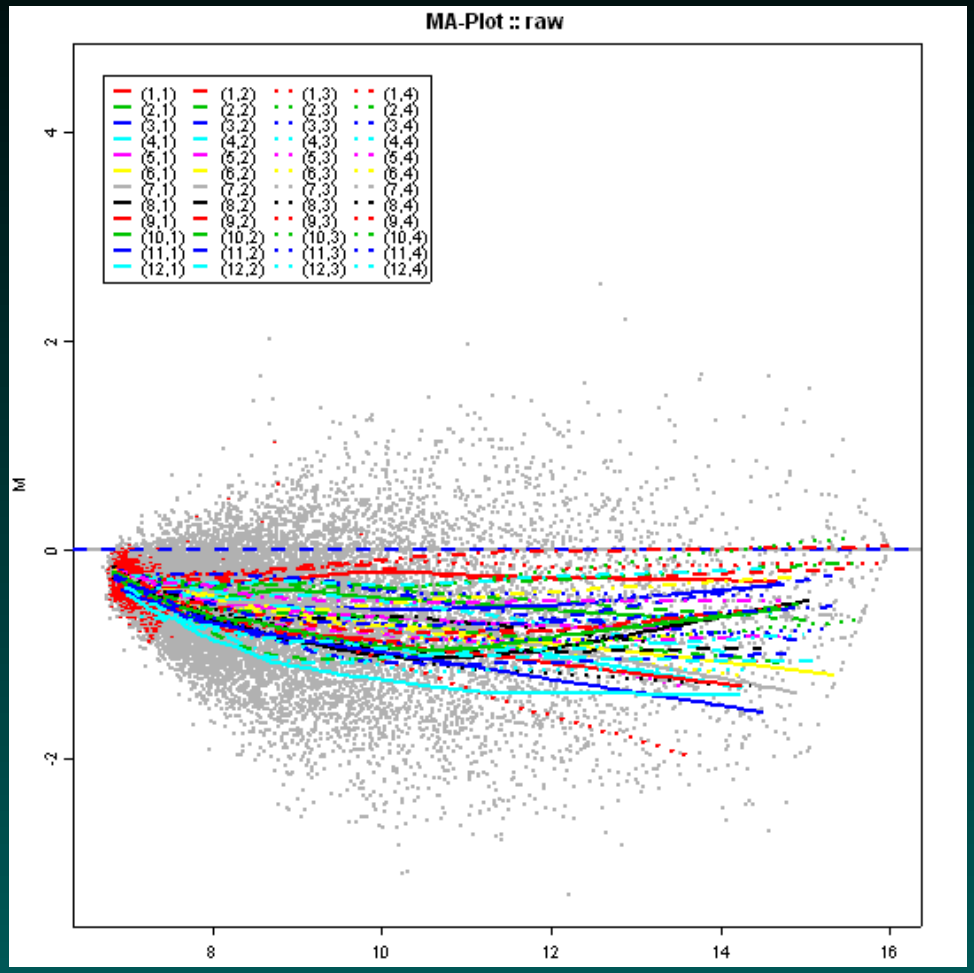
Checking the Data

Ok, now we have the raw data. What do we want to try next? Well, checking array quality would be nice.

```
> maQualityPlots(mraw); # again, works on PC only
save as diagPlot..6Hs.195.1.png
save as diagPlot..6Hs.168.png
save as diagPlot..6Hs.166.png
save as diagPlot..6Hs.187.1.png
save as diagPlot..6Hs.194.png
save as diagPlot..6Hs.243.1.png
```

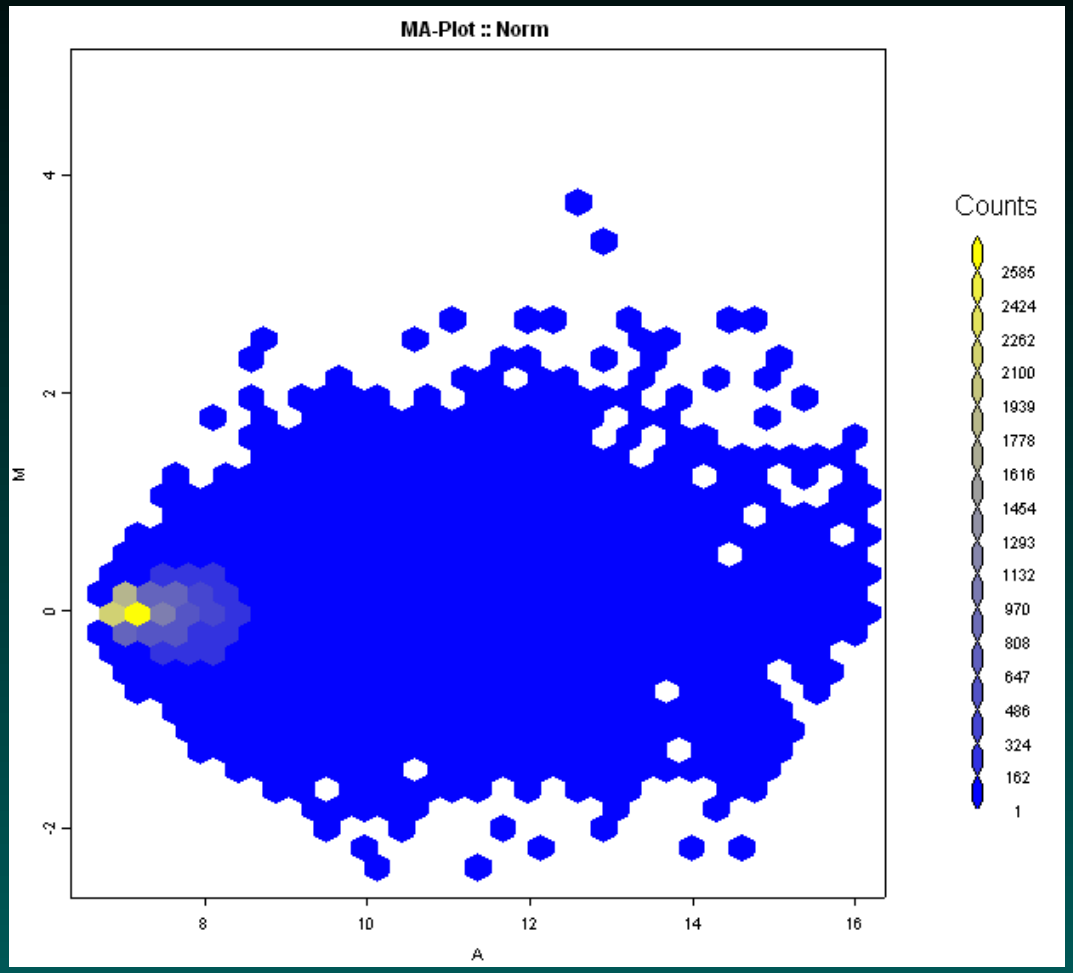
what does this produce? One large png file for each array. This plot has 8 panels...

Panel (a)



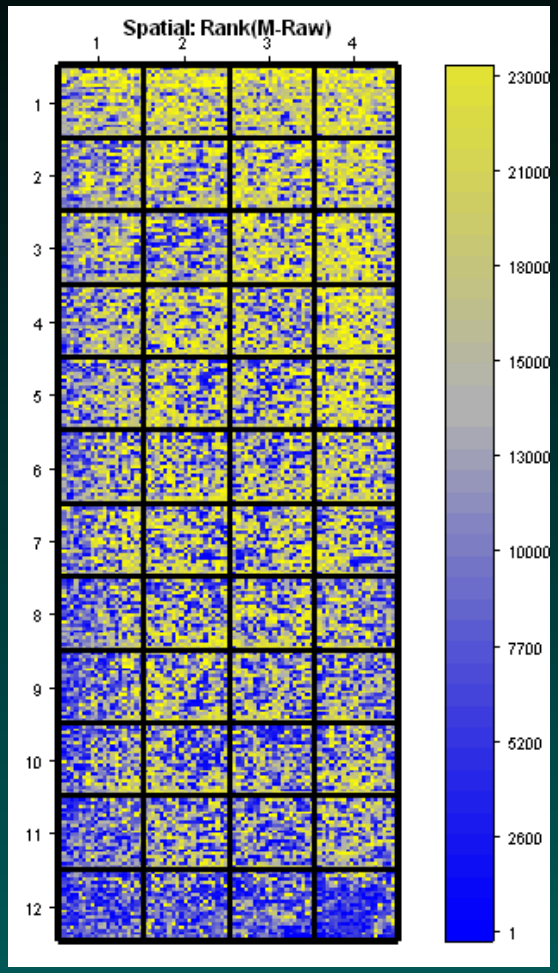
(a) an MA-plot for the raw data, with loess traces for each pin

Panel (b)



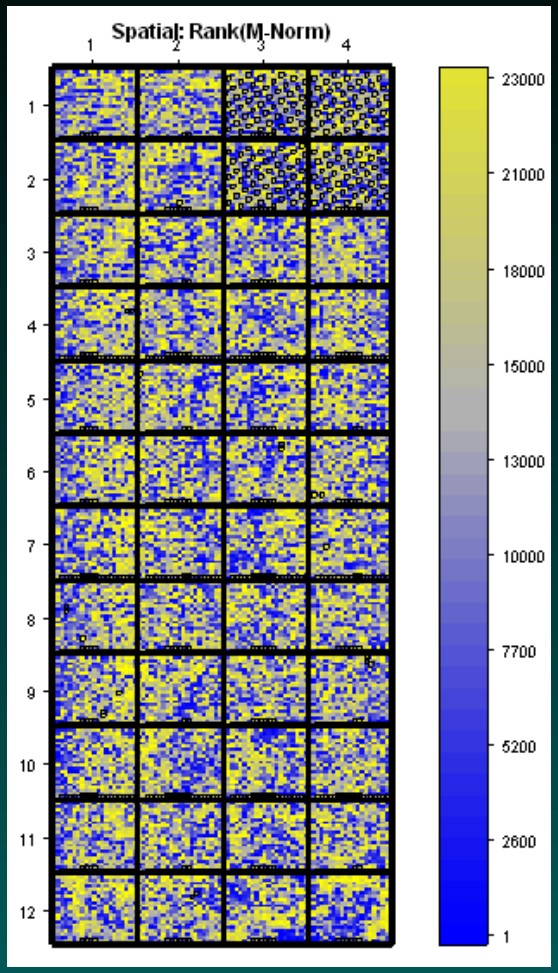
(b) an MA-plot for the data after print-tip loess normalization, displayed using hexbin.

Panel (c)



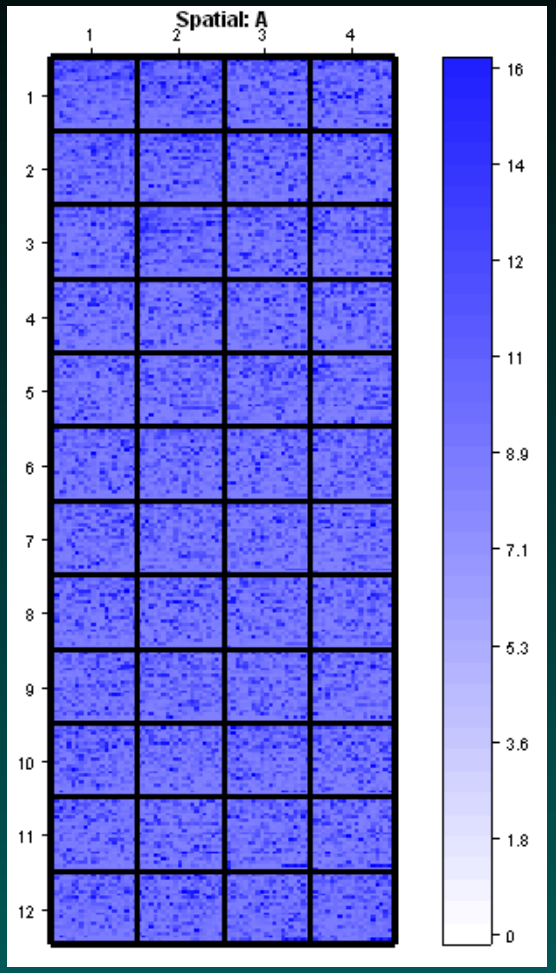
(c) a spatial plot of ranks of the M-Raw differences

Panel (d)



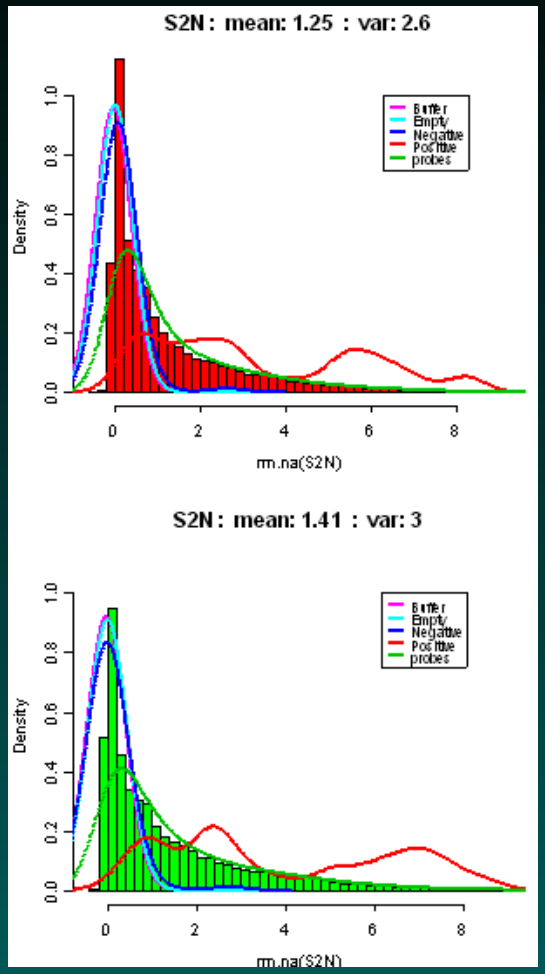
(d) a spatial plot of ranks of the M-Norm differences, with outliers flagged

Panel (e)



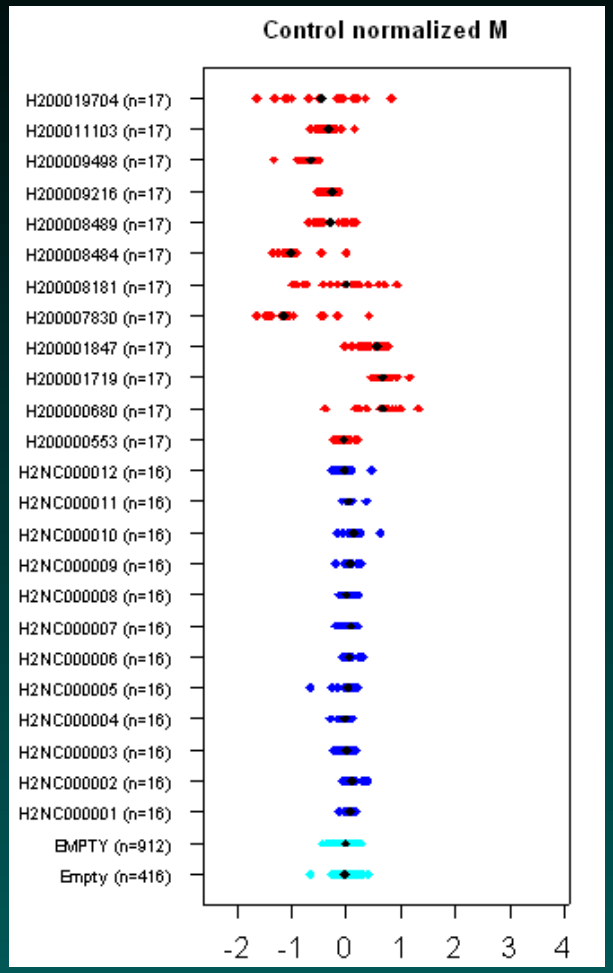
(e) a spatial plot of the A values

Panel (f)



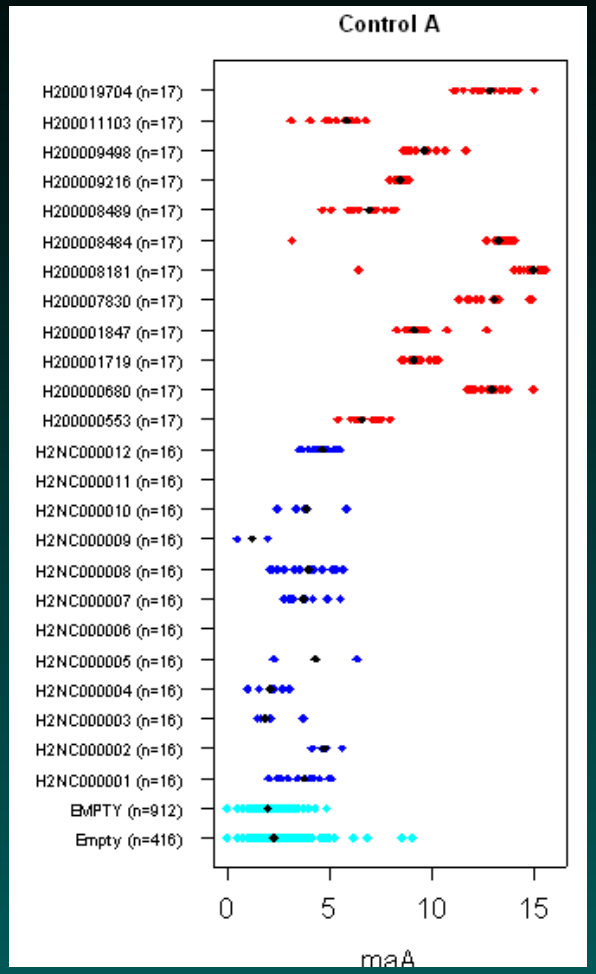
(f) signal to noise distribution plots for each channel (presumably assessed on the raw data)

Panel (g)



(g) M distributions for replicated controls using the normalized values

Panel (h)



(h) A distributions for replicated controls using the normalized values

What next?

Ok, given that the arrays look ok, we'd like to do some numerical contrasts. What needs to be done before we do this?

What next?

Ok, given that the arrays look ok, we'd like to do some numerical contrasts. What needs to be done before we do this?

Go from an marrayRaw object to an marrayNorm object.

```
> normdata <- maNorm(mraw);
```

by default, this will invoke print-tip loess as the processing method.

Exporting the Data

```
write.marray(normdata);
```

This will create a file “maRawResults.xls”, even though the normalized data was used. This will give grid R,C, spot R,C, the spot ID, the gene name, and the associated log ratio values. It presumes that we know which direction the ratios are taken in (it’s Cy5/Cy3).

Using the Data Further

```
library("convert");
mdata <- as(normdata,"exprSet");
```

This would seem to coerce our marrayNorm object into an exprSet, which we can then act upon to get more information. This is partially correct.

The gene names are not retained or passed, so keeping track of the annotation must be done by index value or attached separately.

How was the Data Analyzed?

According to the methods, they worked just with the foreground measurements; no background was subtracted.

Print-tip loess was used to normalize the array data, and log ratios were computed.

Differentially expressed genes were estimated using a linear model (and the limma package). The model:

$$Y_{ij} = \mu + A_i + \epsilon_{ij}$$

The individual (b7+/b7-) log ratio values for each array are expressed in terms of an overall level, a patient effect, and a chip effect. The patient effect lets them deal with replicates intelligently.

More Analysis

For each gene, a “moderated t-test” was performed using an empirical Bayes approach, pooling information about the variance to make the results more stable.

The genes had to be significant at a 0.01 level after a Bonferroni correction, and the mean fold change had to be more than 1.5.

What Other Info was Provided?

Together with the paper, and the data posted to GEO (the layouts of the arrays used, the gpr files, and more information about what the genes are), there was also a supplementary information file giving a MIAME-compliant list of information.

This list was important, as it specified which samples were labeled with Cy5, and which with Cy3. What is recorded in GEO is simply “Channel 1” and “Channel 2”.