GS01 0163
Analysis of Microarray Data

Keith Baggerly and Kevin Coombes
Section of Bioinformatics
Department of Biostatistics and Applied Mathematics
UT M. D. Anderson Cancer Center
kabagg@mdanderson.org
kcoombes@mdanderson.org

31 October 2006

Lecture 18: A Two-Color Case Study

- Case Study Biology
- Getting Data
- Inferences from GPR Files
- Quality Checks
- Further Analysis
- Adventures with the Gene Expression Omnibus

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 $\beta 7+$ and $\beta 7-$ memory T helper cells", BMC Immunology, 5:13.

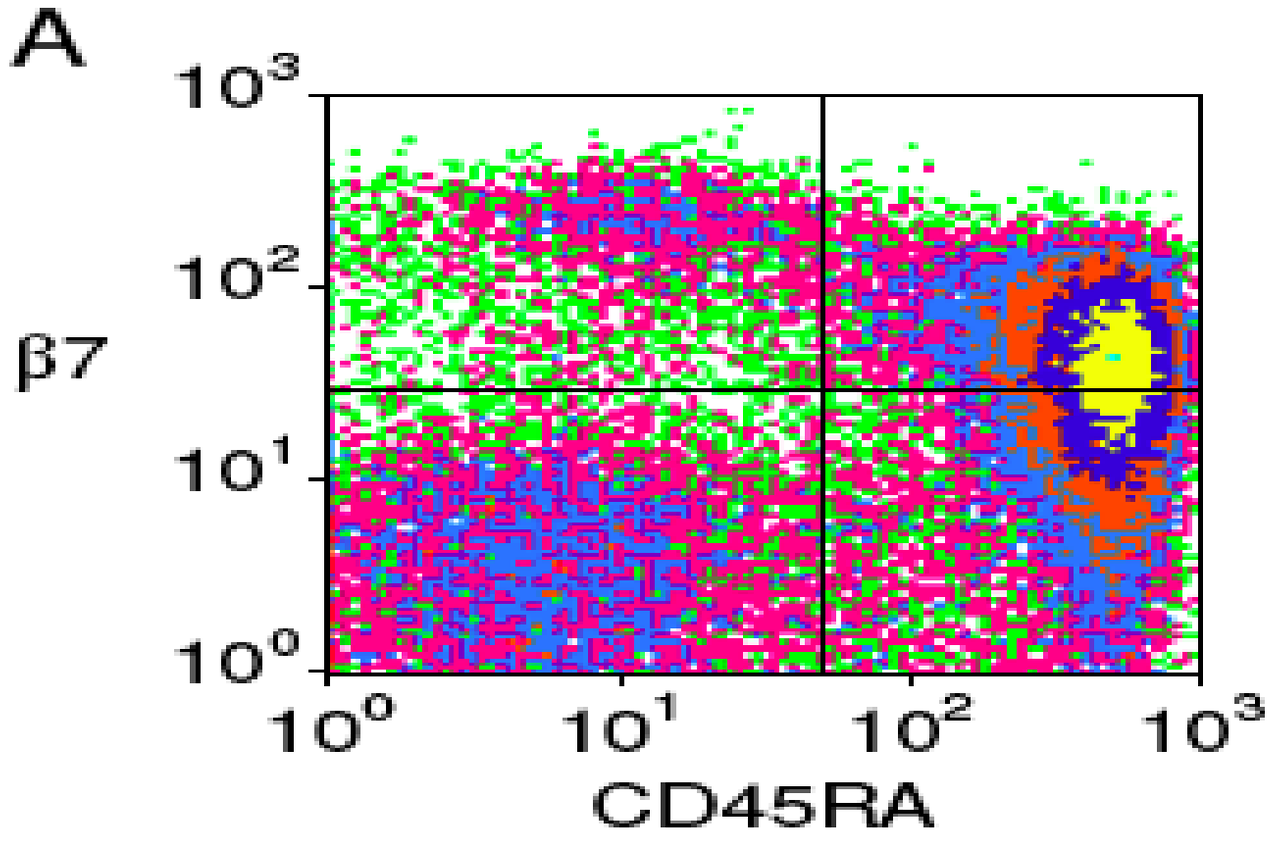
In that paper, they asked whether different types of helper cells were associated with the adhesion or migration of T cells.

How do we Get Cells?

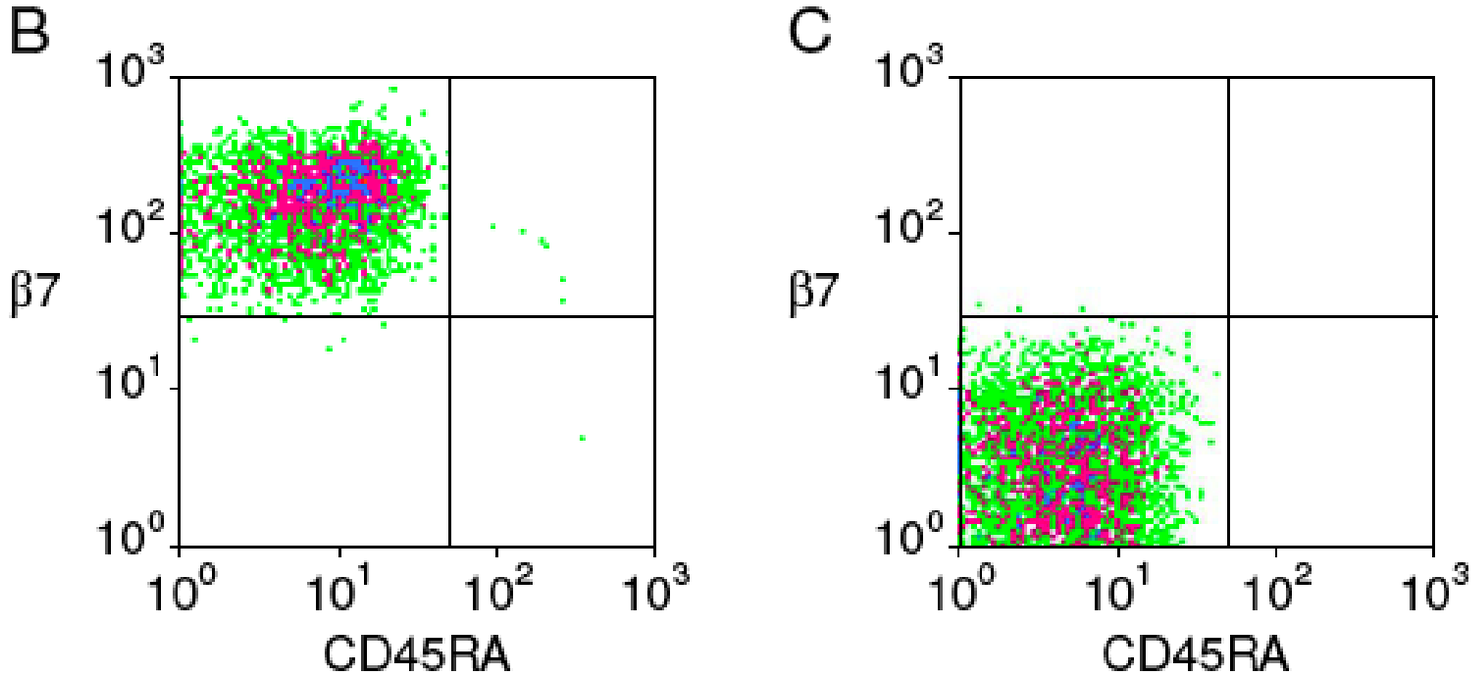
Extract CD4+ T cells, and derive enriched subpopulations that are $\beta 7+$ and $\beta 7-$. Cell subpopulations were obtained using flow cytometry.

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 performed using samples from 9 individuals, so there is a natural data pairing.

Given the pairing, individual arrays were used to contrast the two by hybridizing $\beta 7+$ in one channel and $\beta 7-$ 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 will return to this later.)

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

Next, we 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 (e.g., phenoData information). E.g.:

```
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("Data/TargetBeta7.txt")
> TargetInfo

An object of class "marrayInfo"
@maLabels
[1] "6Hs.195.1.gpr" "6Hs.168.gpr"   "6Hs.166.gpr"
[4] "6Hs.187.1.gpr" "6Hs.194.gpr"   "6Hs.243.1.gpr"

@maInfo
      FileNames Subject ID #  Cy3  Cy5       Hyb buffer
1 6Hs.195.1.gpr            1 b7 - b7 + Ambion Hyb Slide
2   6Hs.168.gpr            3 b7 + b7 - Ambion Hyb Slide
3   6Hs.166.gpr            4 b7 + b7 - Ambion Hyb Slide
4 6Hs.187.1.gpr            6 b7 - b7 + Ambion Hyb Slide
```



```
5   6Hs.194.gpr             8 b7 - b7 + Ambion Hyb Slide
6 6Hs.243.1.gpr            11 b7 + b7 - Ambion Hyb Slide
  Hyb Temp (deg C) Hyb Time (h) Date of Blood Draw
1               55           40          2002.10.11
2               55           40          2003.01.16
3               55           40          2003.01.16
4               55           40          2002.09.16
5               55           40          2003.09.18
6               55           40          2003.01.13
  Amplification  Slide Type Date of Scan
1      R2 aRNA  Aminosilane   2003.07.25
2      R2 aRNA  Aminosilane   2003.08.07
3      R2 aRNA  Aminosilane   2003.08.07
4      R2 aRNA  Aminosilane   2003.07.18
5      R2 aRNA  Aminosilane   2003.07.25
6      R2 aRNA  Aminosilane   2003.08.06
```



```
@maNotes
[1] "Data/TargetBeta7.txt"
```

Getting the Numerical Info

Grab the data from the GPR files:

```
> mraw <- read.GenePix(targets = TargetInfo, path = "Data")
```

```
Reading ...  Data/6Hs.195.1.gpr
Reading ...  Data/6Hs.168.gpr
Reading ...  Data/6Hs.166.gpr
Reading ...  Data/6Hs.187.1.gpr
Reading ...  Data/6Hs.194.gpr
Reading ...  Data/6Hs.243.1.gpr
```


Mac errors?

Note: this works on a PC. On a Mac laptop, Keith reported 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, fixed,
     useBytes)
2: input string 32 is invalid in this locale in:
  grep(pattern, x, ignore.case, extended, value, fixed,
   useBytes)
```


What Can be Inferred?

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

We look at the individual slots.

```
> slotNames(mraw)
```

```
[1] "maRf"       "maGf"       "maRb"       "maGb"
[5] "maW"        "maLayout"   "maGnames"   "maTargets"
[9] "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. We will extract these to 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(zedL)
> zedLGR <- maGridRow(zedL)
> zedLGC <- maGridCol(zedL)
> zedLcoords <- cbind(zedLGR, zedLGC, zedLSR, zedLSC)
> zedLcoords[c(1:2, 20:22), ]
```

```
      zedLGR zedLGC zedLSR zedLSC
[1,]      1      1      1      1
[2,]      1      1      1      2
[3,]      1      1      1     20
[4,]      1      1      1     21
[5,]      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 Blood Draw  Date of Scan
1         2002.10.11    2003.07.25
2         2003.01.16    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.	Max.	NA's
6Hs.195.1.gpr	-6.13	-1.00	-0.52	-0.50	-0.08	5.95	3415
6Hs.168.gpr	-7.08	-0.80	-0.21	-0.23	0.34	5.19	2839
6Hs.166.gpr	-7.07	-1.25	-0.64	-0.62	-0.02	6.15	3440
6Hs.187.1.gpr	-9.81	-0.92	-0.60	-0.55	-0.25	5.00	2942
6Hs.194.gpr	-5.93	0.00	0.44	0.53	0.90	7.74	6090
6Hs.243.1.gpr	-6.38	-1.13	-0.69	-0.64	-0.21	7.05	2227

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"
[1] "H200000297" "H200000303"
                    ID
H200000297 H200000297
H200000303 H200000303
                  Name
H200000297 OVGP1 - Oviductal glycoprotein 1, 120kD (mucin 9,
H200000303 TAF1 - TAF1 RNA polymerase II, TATA box binding pr
[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

So, what if you are 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!

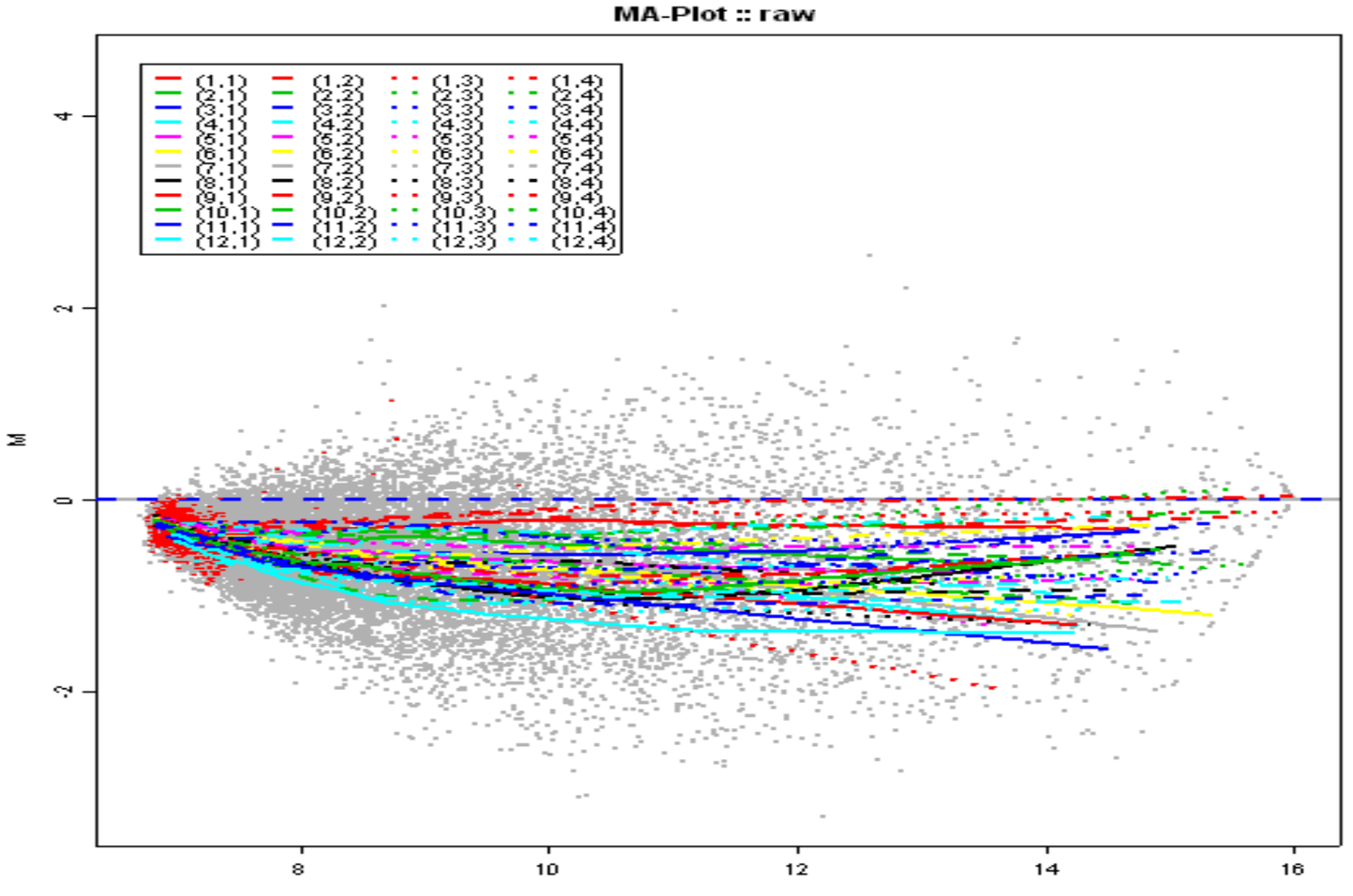
Checking the Data

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
```

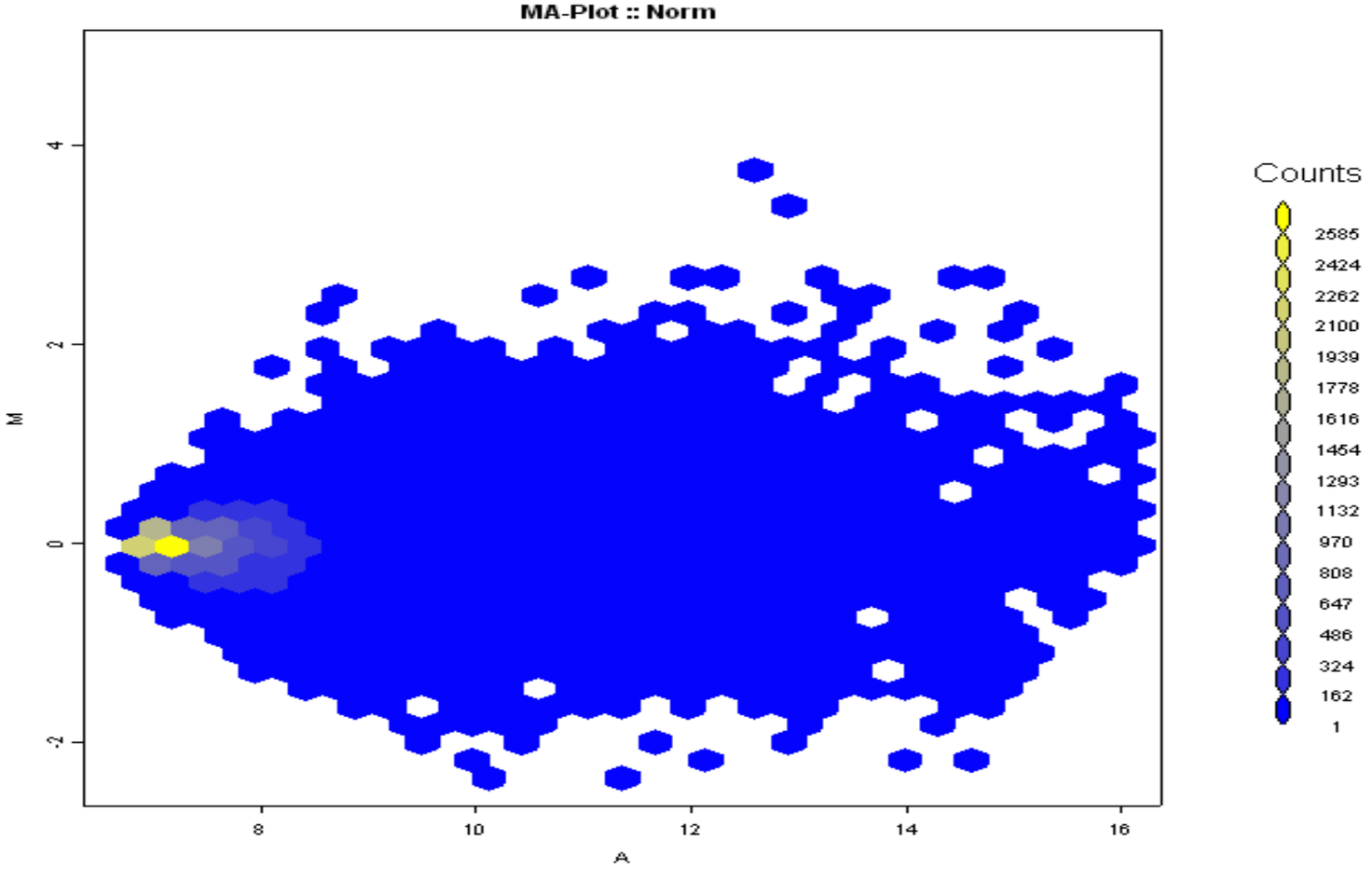
What does this produce? One large png file for each array. This plot has 8 panels...

Panel (a)



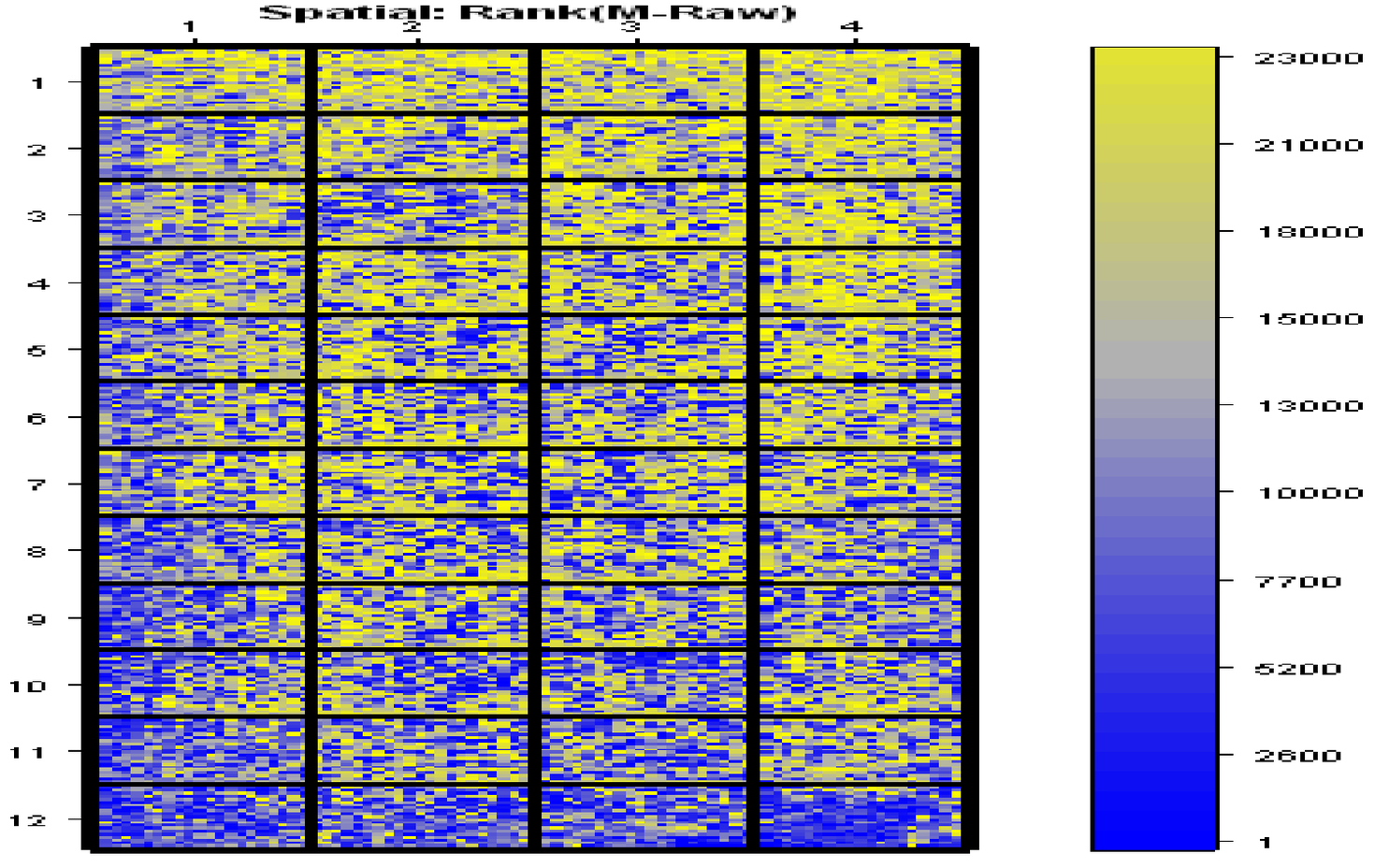
(a) an MA-plot for the raw data, with loess traces for each pin

Panel (b)



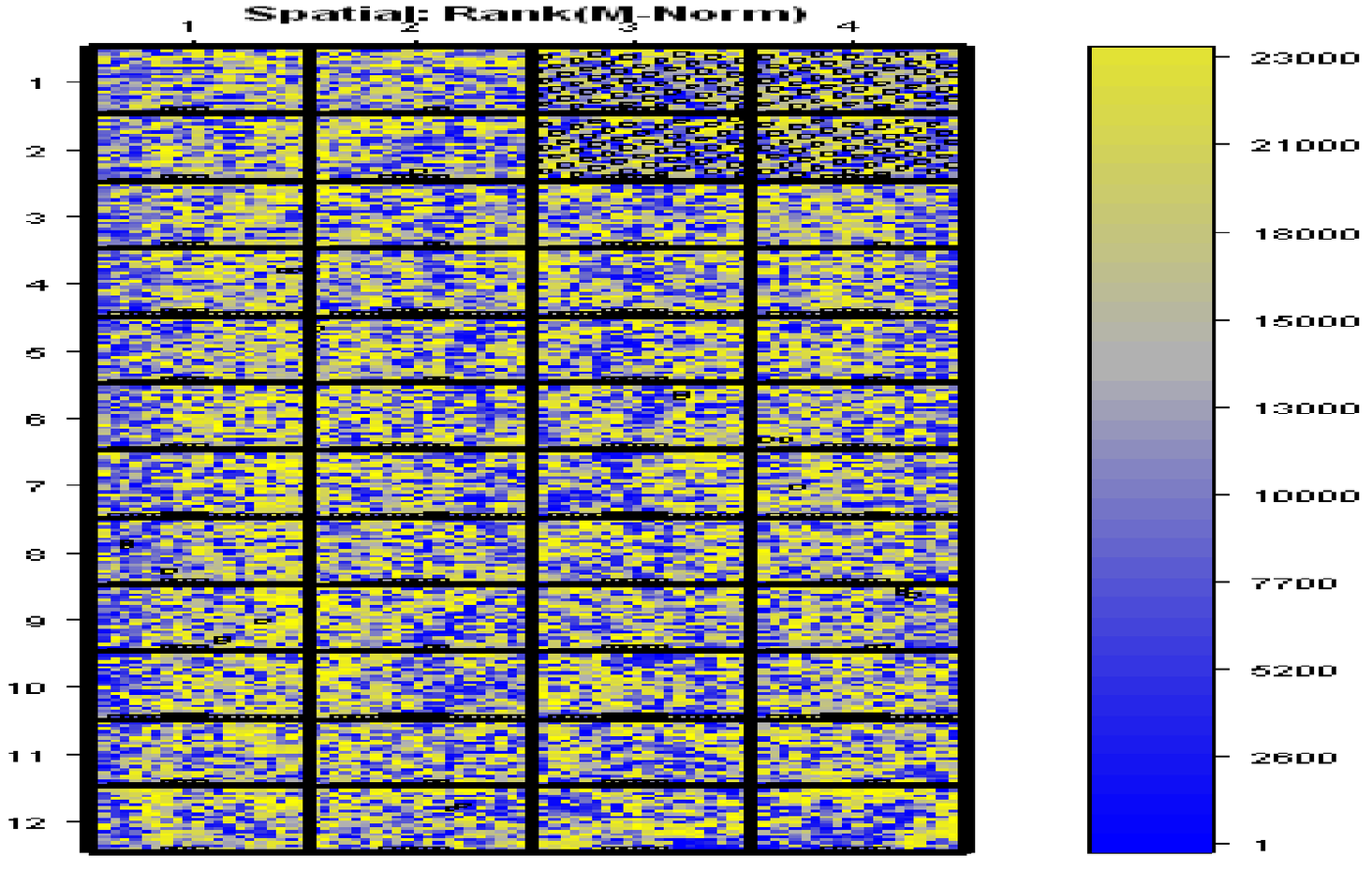
(b) an MA-plot for the data after print-tip loess normalization, displayed using hexbin.

Panel (c)



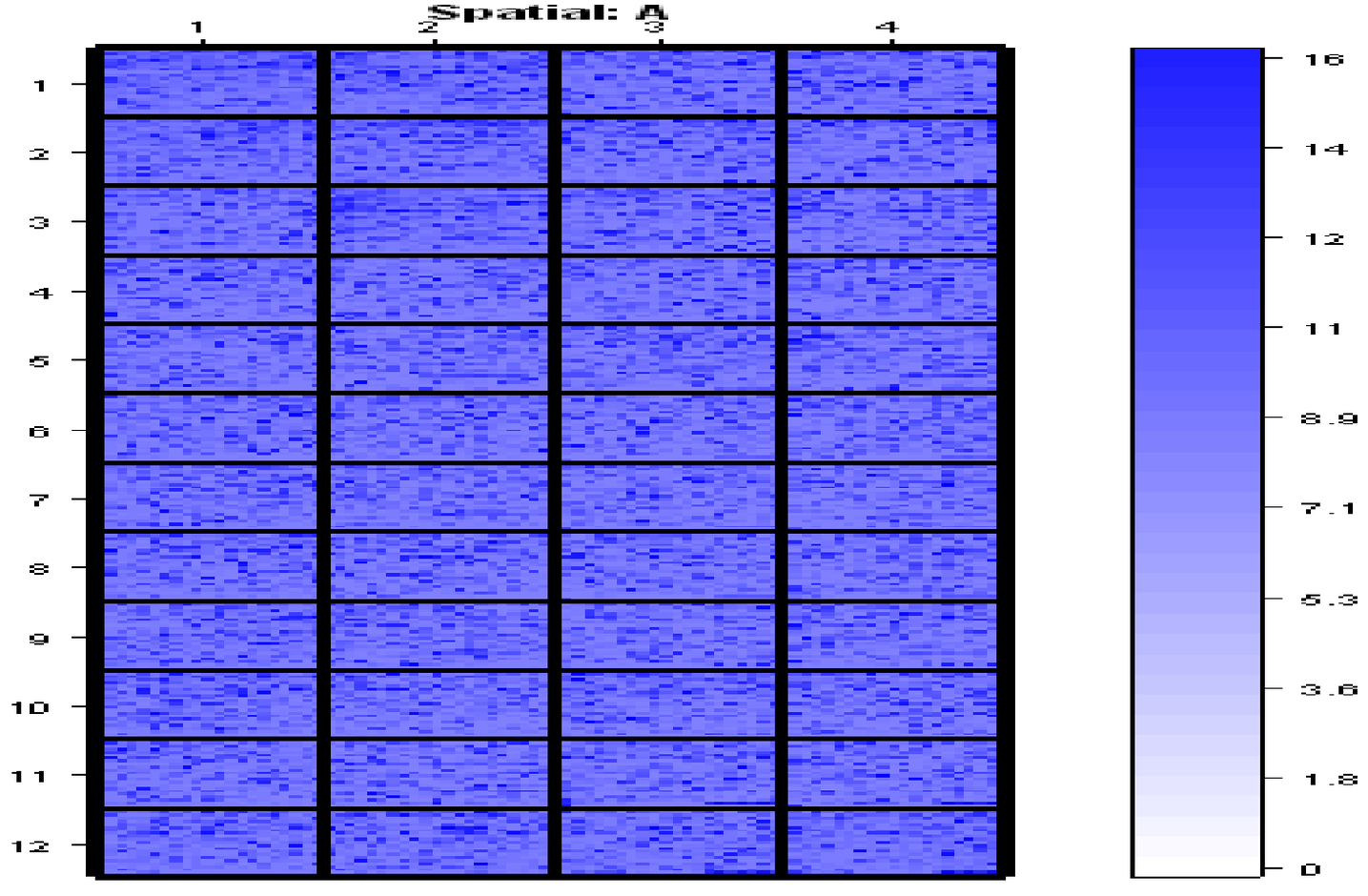
(c) a spatial plot of ranks of the M-Raw differences

Panel (d)



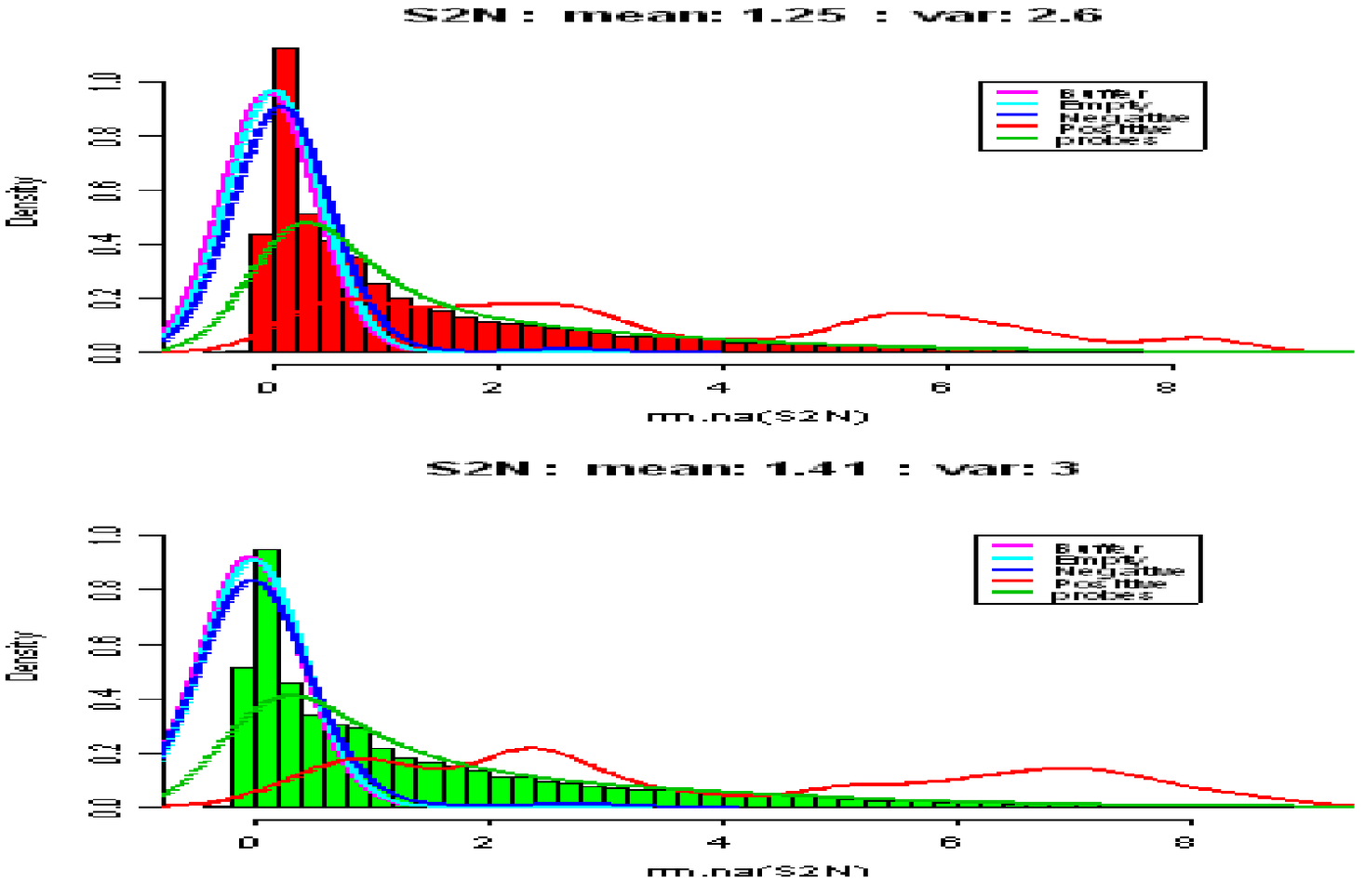
(d) a spatial plot of ranks of the M-Norm differences, with outliers flagged

Panel (e)



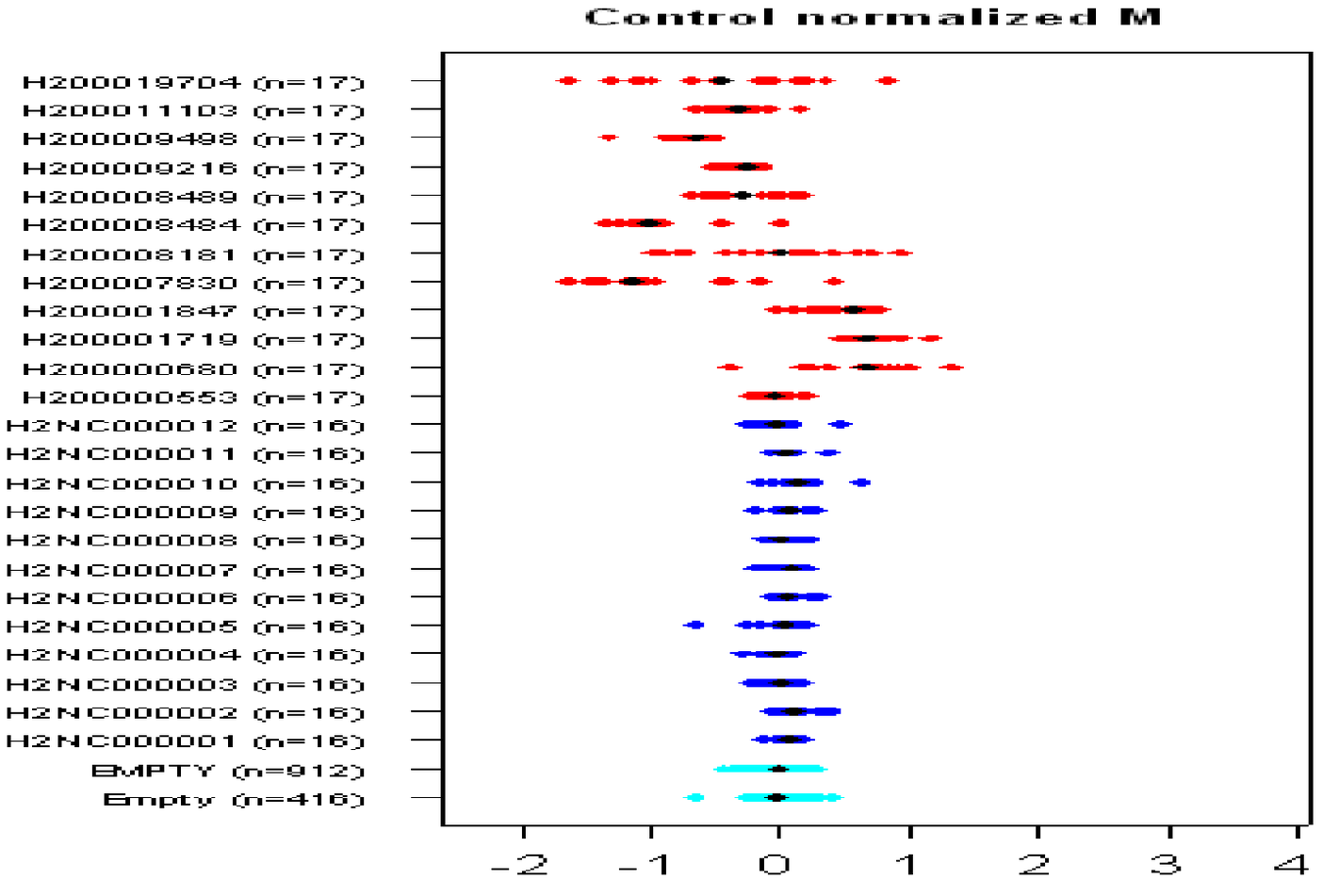
(e) a spatial plot of the A values

Panel (f)



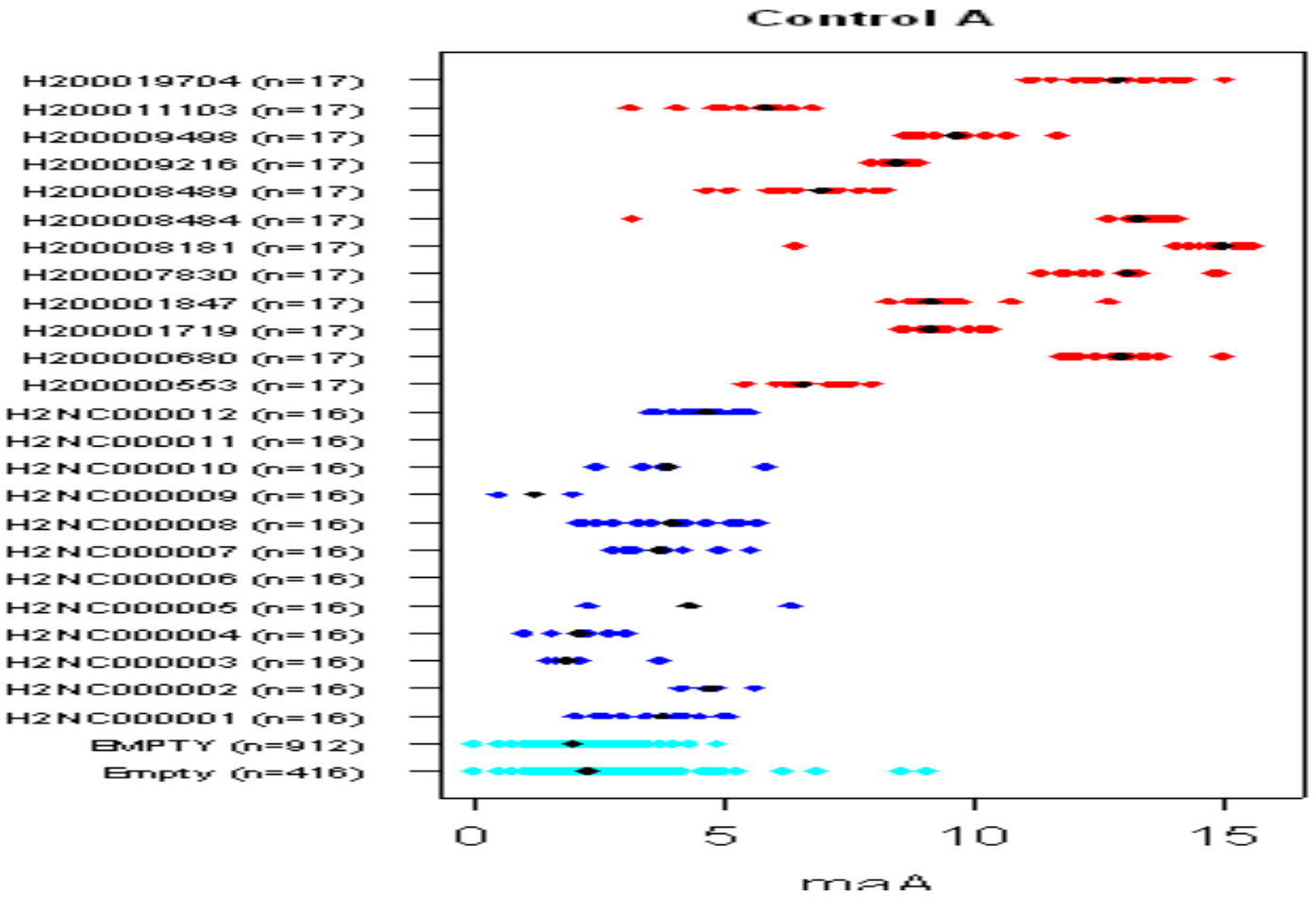
(f) signal to noise distribution plots for each channel (presumably assessed on the raw data)

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 would 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)
```

```
NULL
```

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 (in, fact, 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?

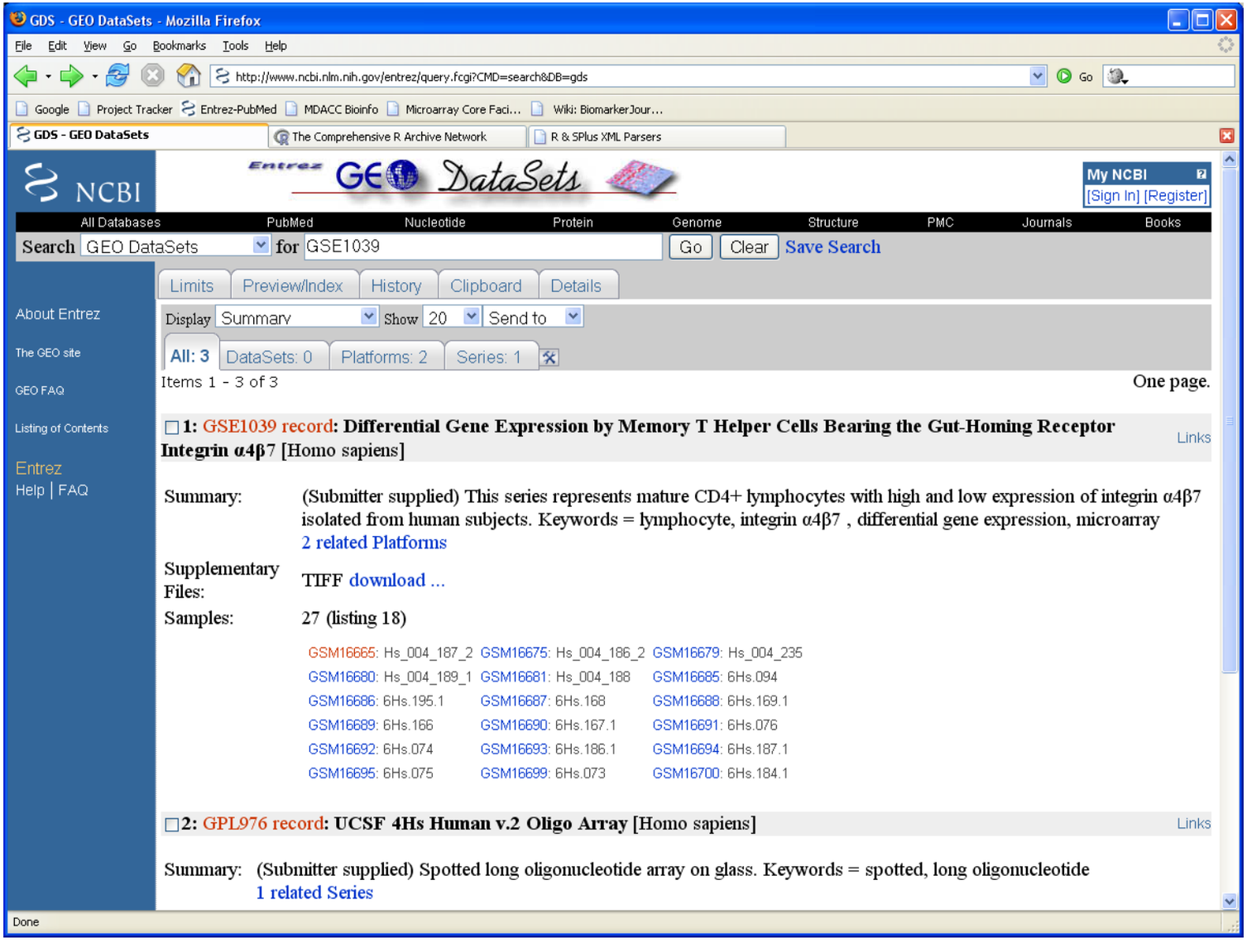
Together with the paper, and the data posted to GEO (the layouts of the arrays used, the gpr files, and more information about what the genes are), there was also a supplementary information file giving a MIAME-compliant list of information.

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".

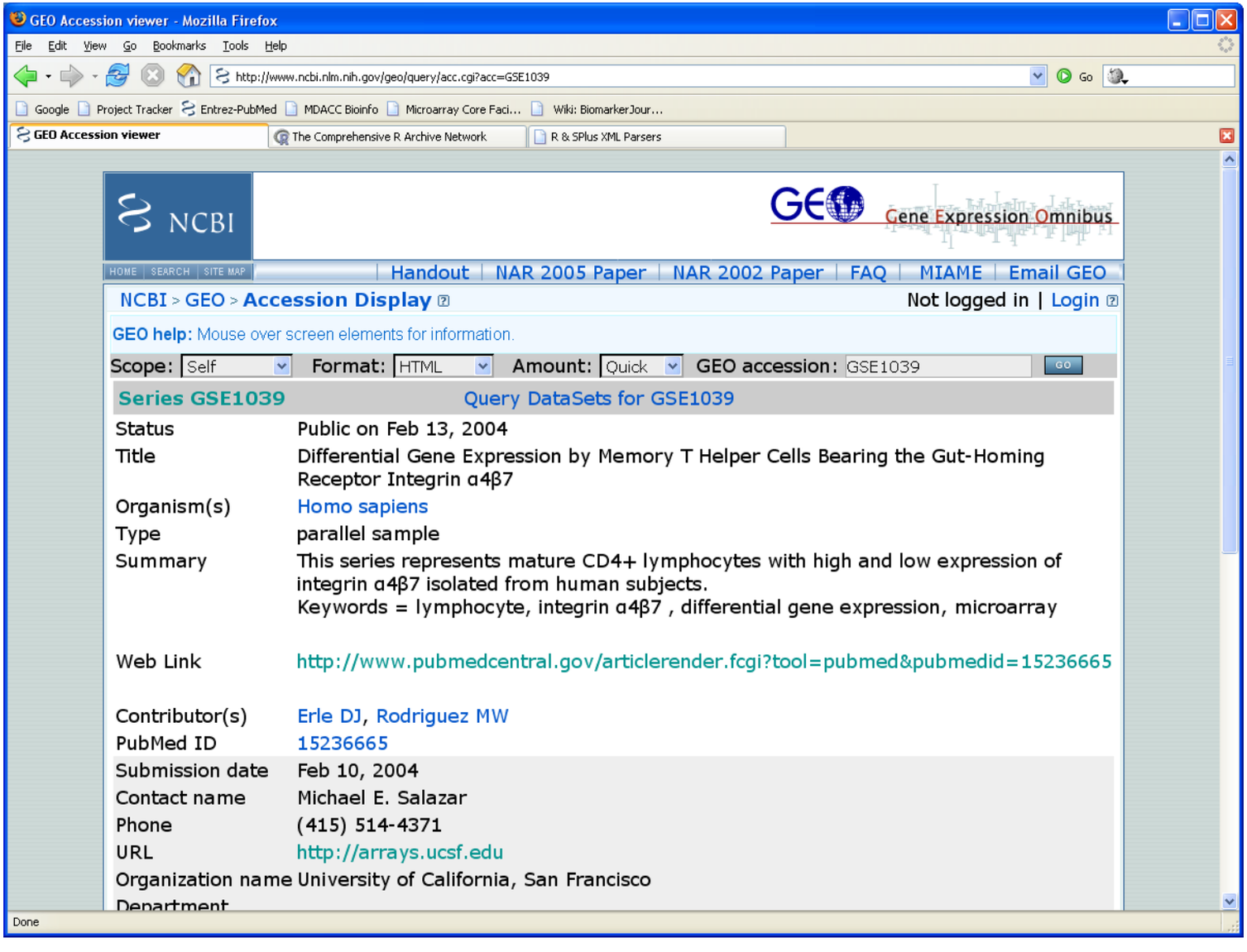
Adventures with the Gene Expression Omnibus

I went back to GEO to find the full data set, to see how easy (ha!) it would be to get the whole thing into R. Maybe by Thursday I will have figured this all out. In the mean time, you get to share my confusion/frustration....

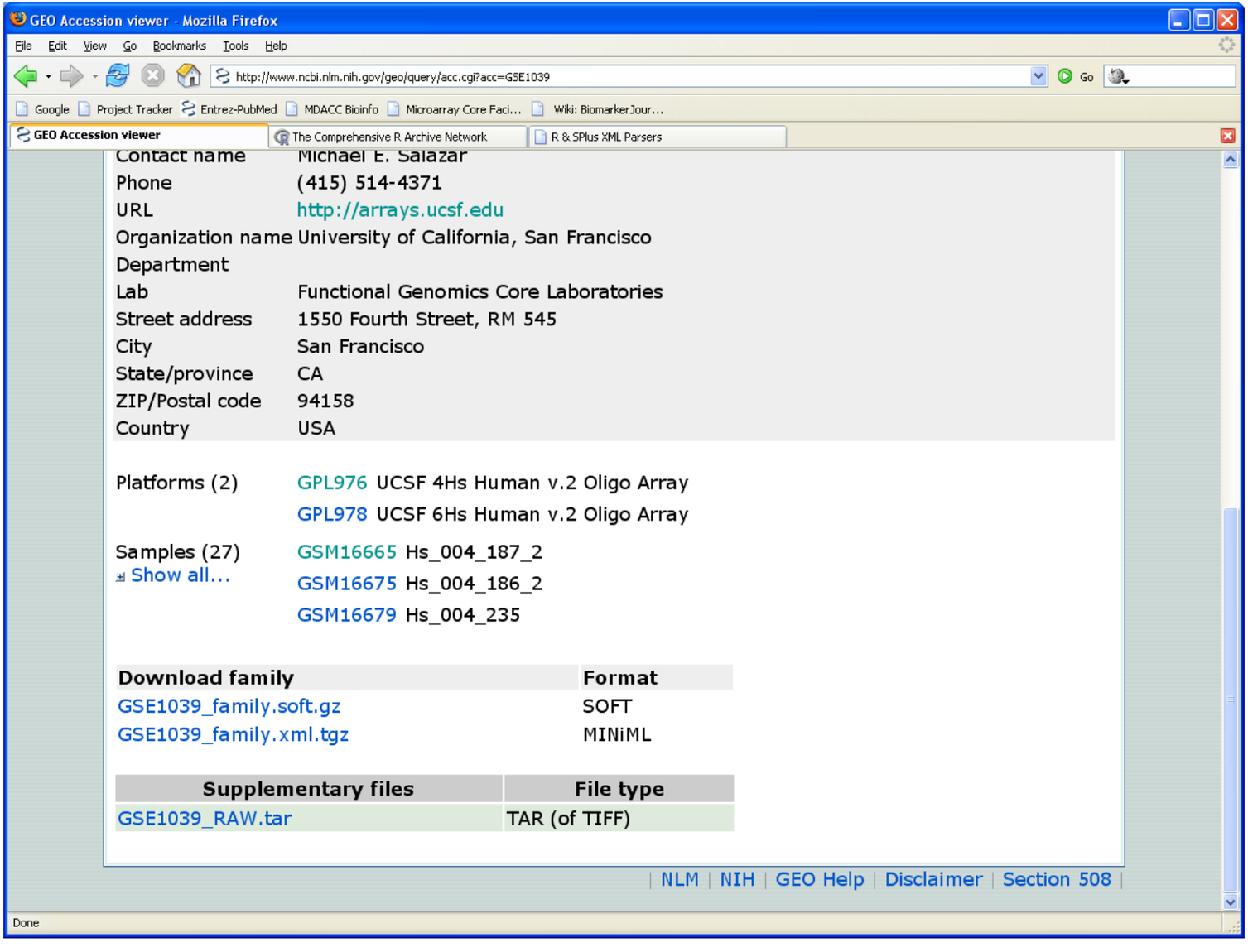
Searching for the Data Set



The Source Page for GSE1039



Scroll Down to FInd the Link

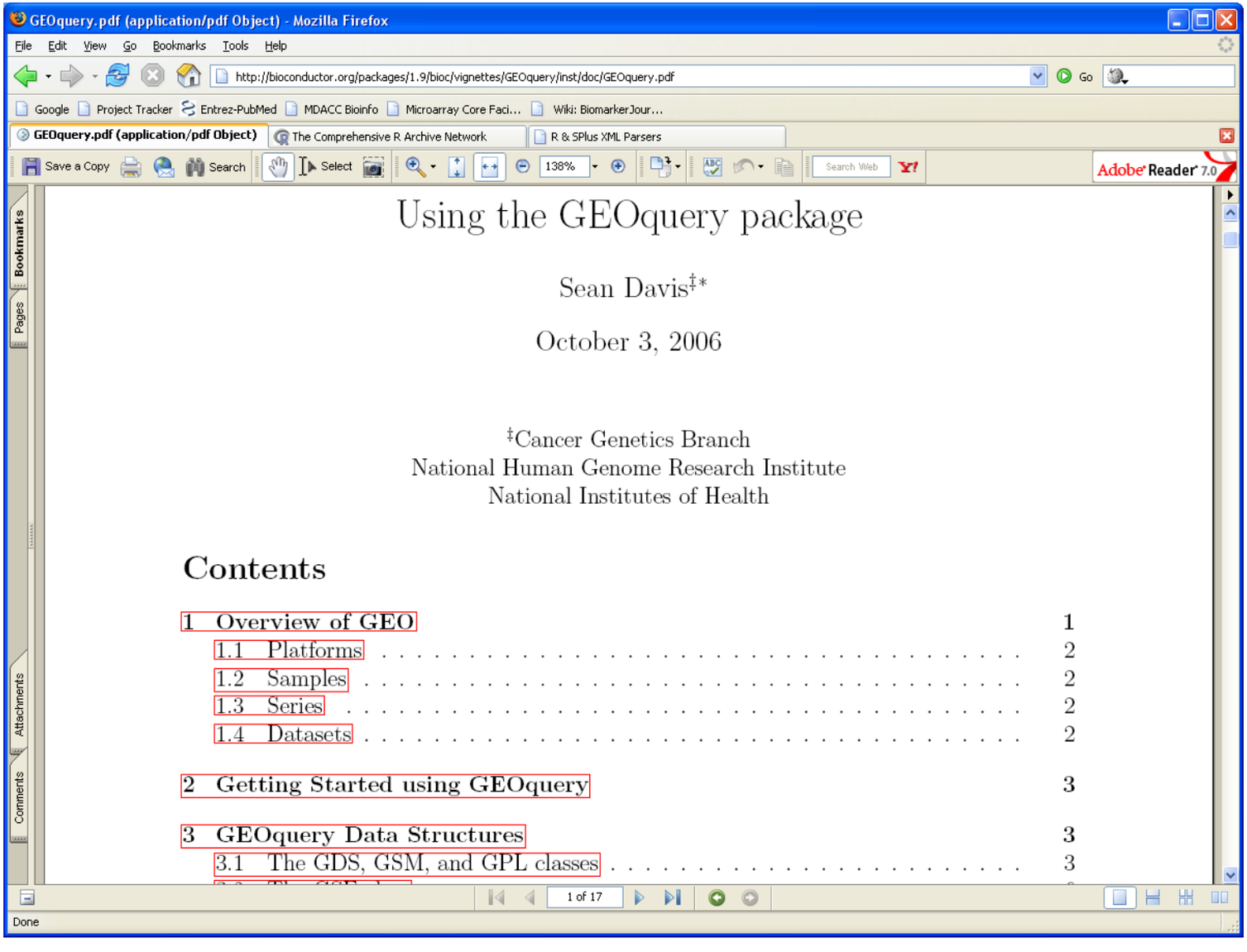


File Formats

The data set is available in two different file formats:

1. SOFT format
   - One big file that clunps everything together, with special characters separating the pieces
   - A google search for "BioConductor SOFT" uncovers the `GEOquery` package to handle SOFT files
2. MINiML format
   - A "tarball" of files, with the documentation in one XML file and everfything else in tab-separated-values format with no headers
   - Another google search turns up the `XML` package for R

GEOquery


GEOquery.pdf (application/pdf Object) - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://bioconductor.org/packages/1.9/bioc/vignettes/GEOquery/inst/doc/GEOquery.pdf
Google Project Tracker Entrez-PubMed MDACC Bioinfo Microarray Core Faci... Wiki: BiomarkerJour...
GEOquery.pdf (application/pdf Object)
The Comprehensive R Archive Network
R & SPlus XML Parsers
Save a Copy Search Select 138%
Adobe Reader 7.0
Bookmarks Pages Attachments Comments
Using the GEOquery package
Sean Davis‡*
October 3, 2006
‡Cancer Genetics Branch
National Human Genome Research Institute
National Institutes of Health
Contents
1 Overview of GEO 1
1.1 Platforms 2
1.2 Samples 2
1.3 Series 2
1.4 Datasets 2
2 Getting Started using GEOquery 3
3 GEOquery Data Structures 3
3.1 The GDS, GSM, and GPL classes 3
1 of 17
Done

GEOquery

So, I installed the `GEOquery` package, and then loaded it.

```
> require(GEOquery)
```

```
[1] TRUE
```

If you just pass the "GSE1039" identifier into the `getGEO` function, it will download the file from the NCBI and start processing it. Since I had already downlaoded it myself, I used the `filename` parameter to the function.

```
> gse1039 <- getGEO(filename = "GSE1039_family.soft")
```

Then I waited. And waited. After more than an hour-and-half, I finally saw some results:

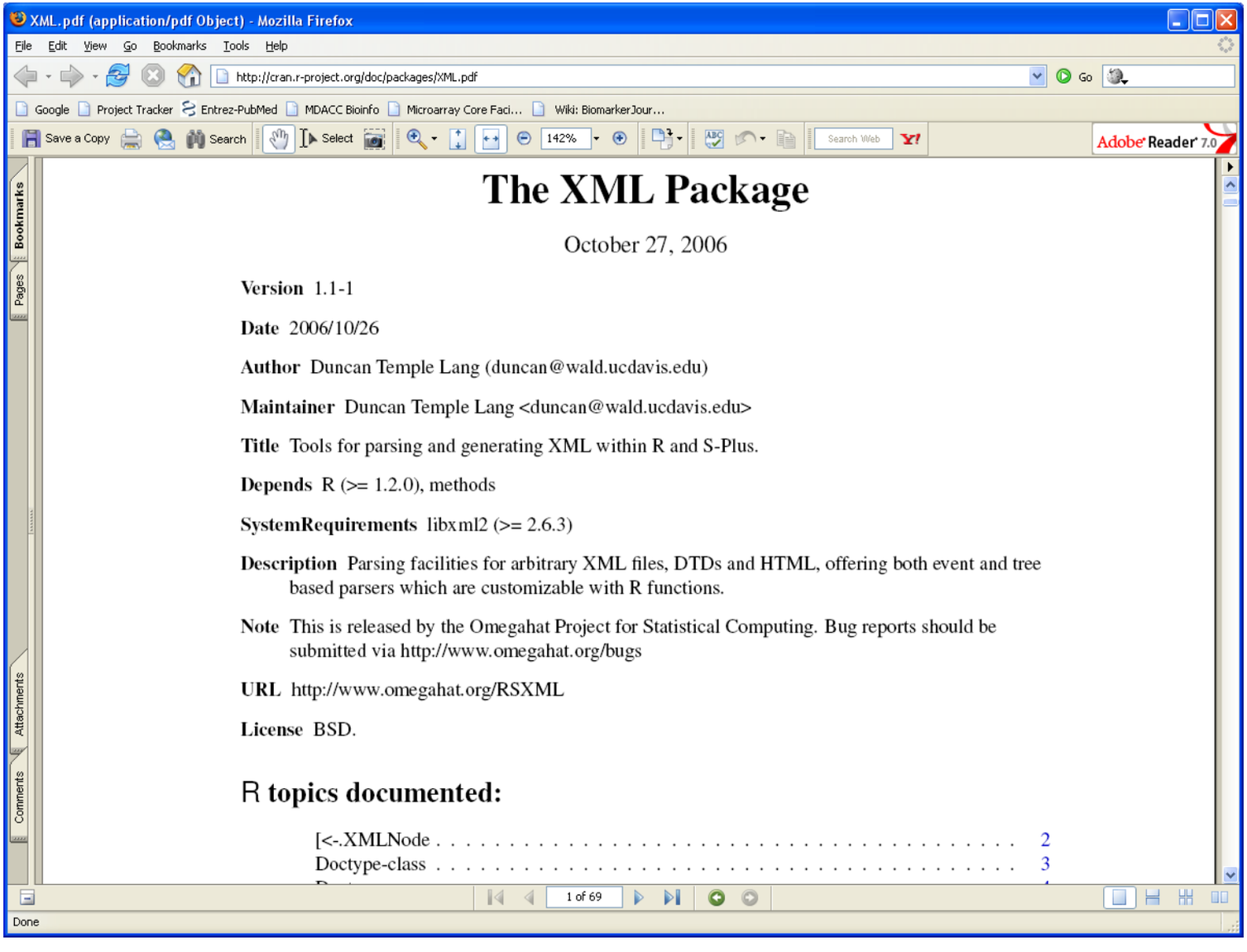

```
Parsing....
^PLATFORM = GPL976
^SAMPLE = GSM16675
^SAMPLE = GSM16679
^SAMPLE = GSM16680
^SAMPLE = GSM16681
^SAMPLE = GSM16685
^SAMPLE = GSM16686
^SAMPLE = GSM16687
^SAMPLE = GSM16688
^SAMPLE = GSM16689
^SAMPLE = GSM16690
^SAMPLE = GSM16691
^SAMPLE = GSM16692
^SAMPLE = GSM16693
^SAMPLE = GSM16694
```



```
^SAMPLE = GSM16695
^SAMPLE = GSM16699
^SAMPLE = GSM16700
^SAMPLE = GSM16704
^SAMPLE = GSM16705
^SAMPLE = GSM16706
^SAMPLE = GSM16719
^SAMPLE = GSM16720
^SAMPLE = GSM16724
^SAMPLE = GSM16725
^SAMPLE = GSM16726
^SAMPLE = GSM16727
```

While this was going on, I decided to check out the XML approach ...

XML

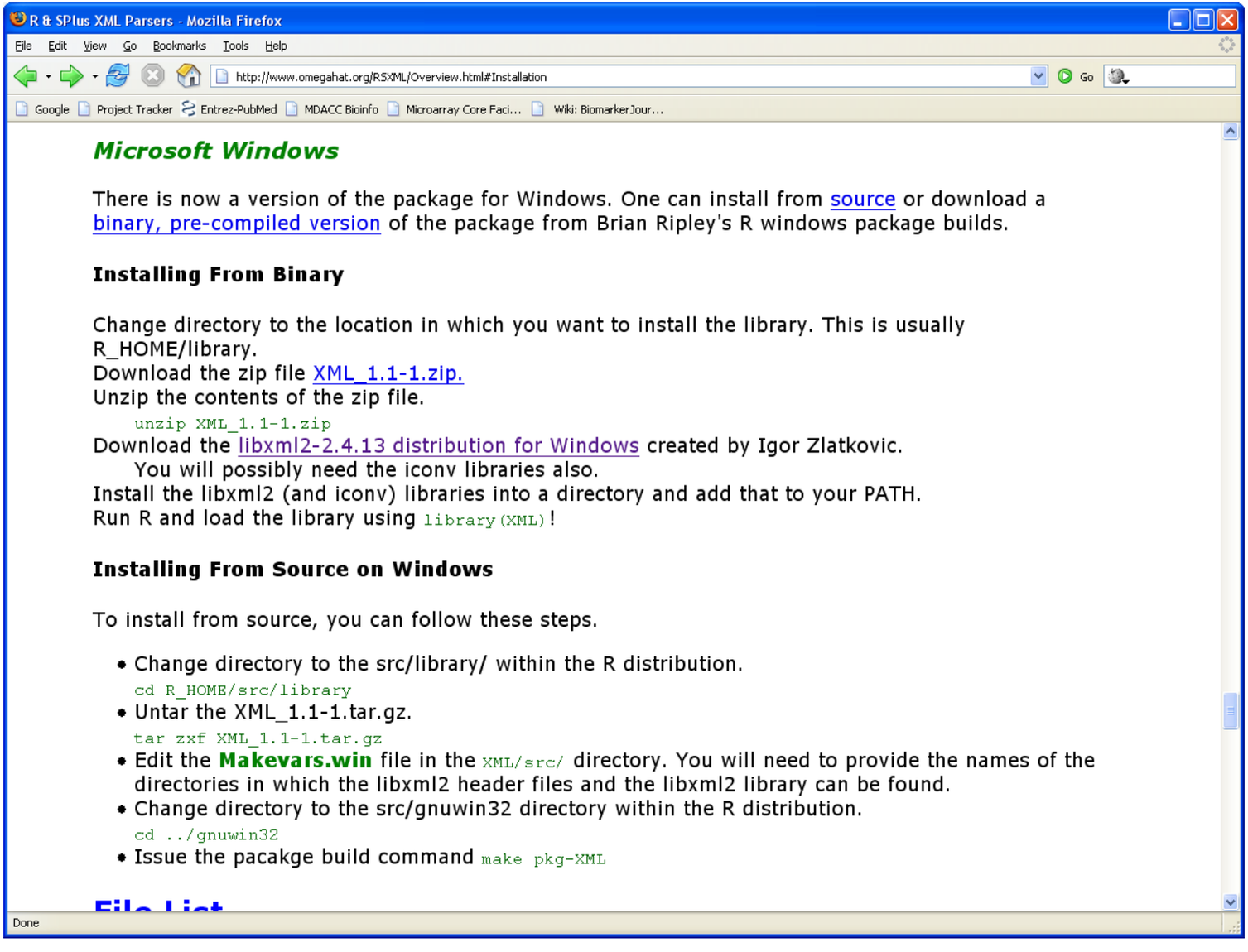

XML.pdf (application/pdf Object) - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://cran.r-project.org/doc/packages/XML.pdf
Google Project Tracker Entrez-PubMed MDACC Bioinfo Microarray Core Faci... Wiki: BiomarkerJour...
Save a Copy Search Select 142% Search Web Adobe Reader 7.0
Bookmarks Pages Attachments Comments
The XML Package
October 27, 2006
Version 1.1-1
Date 2006/10/26
Author Duncan Temple Lang (duncan@wald.ucdavis.edu)
Maintainer Duncan Temple Lang <duncan@wald.ucdavis.edu>
Title Tools for parsing and generating XML within R and S-Plus.
Depends R (>= 1.2.0), methods
SystemRequirements libxml2 (>= 2.6.3)
Description Parsing facilities for arbitrary XML files, DTDs and HTML, offering both event and tree based parsers which are customizable with R functions.
Note This is released by the Omegahat Project for Statistical Computing. Bug reports should be submitted via http://www.omegahat.org/bugs
URL http://www.omegahat.org/RSXML
License BSD.
R topics documented:
[<-.XMLNode 2
Doctype-class 3
1 of 69
Done

Adventures with XML, Part 1

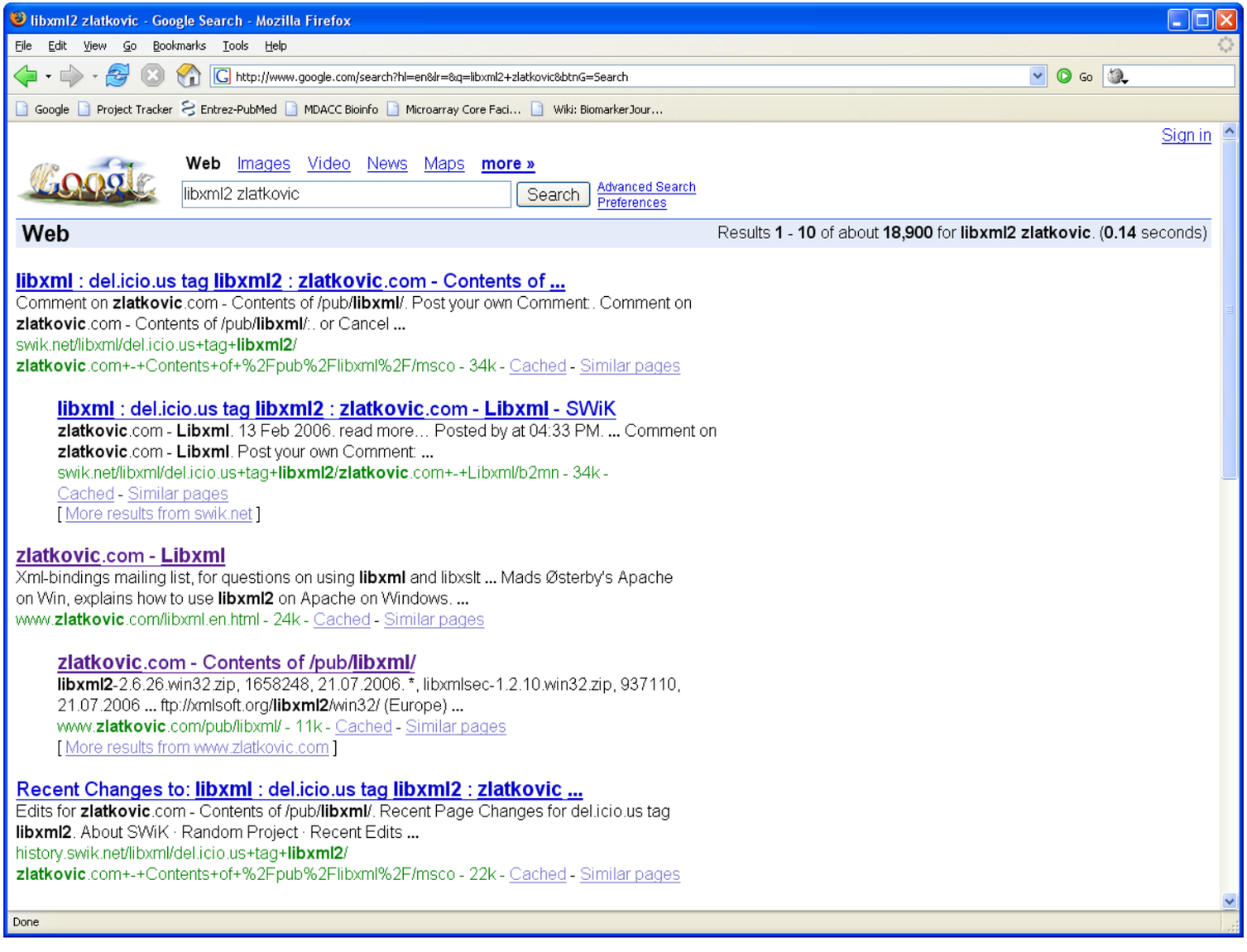
After installing the XML package from CRAN, I foolishly went ahead and tried to use it. Of course, this naive attempt failed. Unlike most packages, this one is decidely not self-contained. It required a separate set of files to parse XML files, which I learned by going back to the original web site form the developer of the package.

R XML Home Page

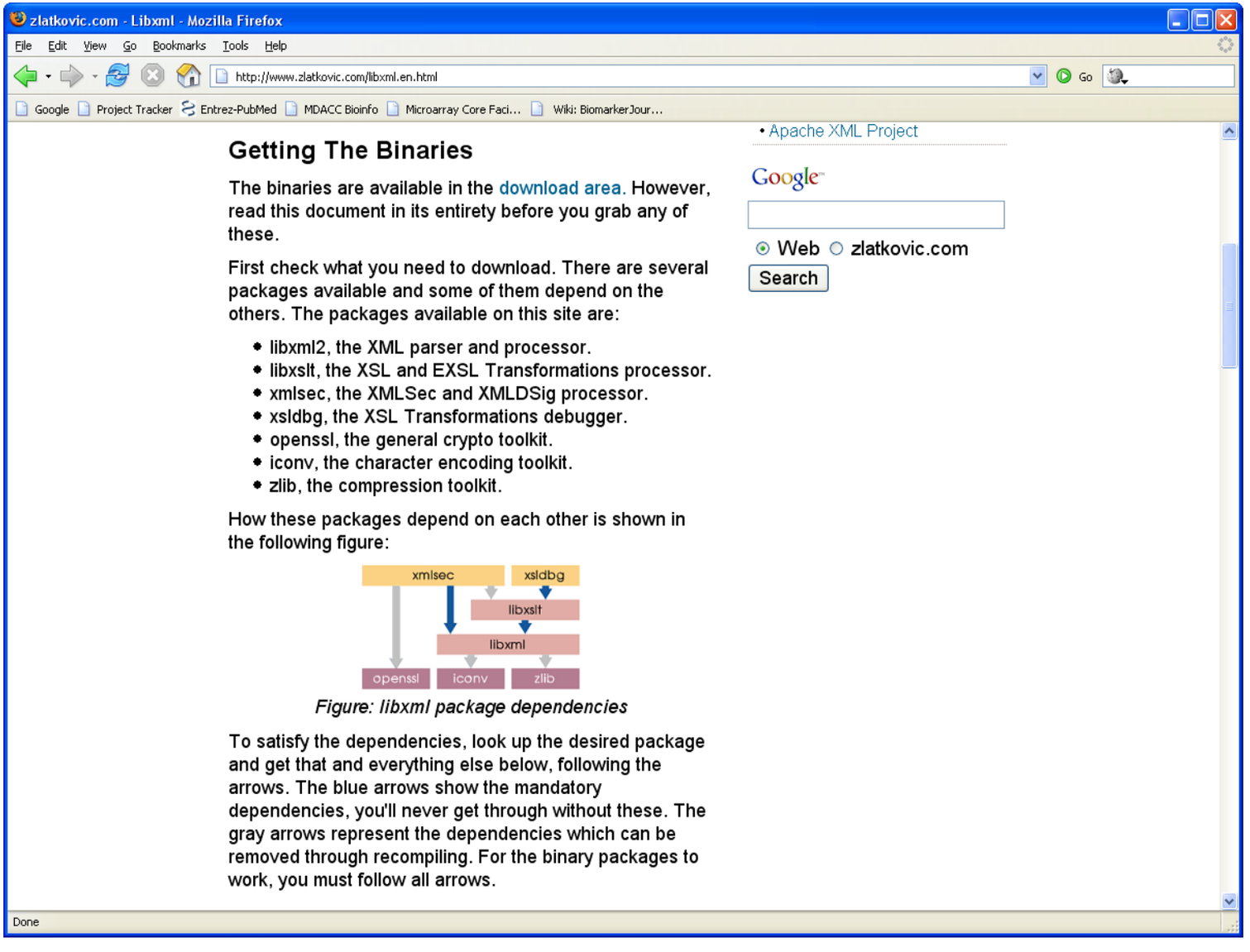
The XML Installation Instructions



The Link to libxml2 Was Broken...



This Time, We Read the Instructions First...



Adventures with XML, Part 2

```
> require(XML)

[1] TRUE

> mytree <- xmlTreeParse("GSE1039/GSE1039_family.xml")

GSE1039/GSE1039_family.xml:242: parser error : Input is not
 proper  UTF-8, indicate encoding !
Bytes: 0xB5 0x6D 0x20 0x6F
        <Description>the X-coordinate in ţm of the center of
                                                ^
```


UTF-8 Encoding Ain't What it Used To Be

After more trials and tribulations, we learned that the XML file lied when it claimed to be "UTF-8" encoded. It included some characters for the Greek letter "mu" and some superscripts (as in R^2) that were not encoded propoerly, breaking the XML parser. So, I fired up my trusty copy of emacs and did a global "search-and-replace" to remove the offending characters. Then we get

```
> mytree <- xmlTreeParse("GSE1039/GSE1039_family2.xml")
```

with no error messages. But I know have an object that represents a parse tree, and not enough time or energy to figure out how to use it....

