Software for affy data analysis

Bioconductor, AffyExtensions and RMAExpress

Ben Bolstad

June 18, 2003

bolstad@stat.berkeley.edu

Biostatistics, University of California, Berkeley

Outline/Topics for discussion

- Data - CDF and CEL files.
- affy/Bioconductor
- AffyExtensions
- RMAExpress

First a few definitions

- **Probe level** pertaining to data at the level of PM and MM. The intensities come directly from the CEL file.
- **Probeset level** data that has in some way been summarized by combining probe information possibly across arrays.

Data - CEL files

For an experiment you will have a CEL file for each chip that you are dealing with (so you will have many of these files). A Cel file contains

- header information with chip size, various information about the experiment, name of CDF file.
- Probe intensity values for each probe on the chip
- OUTLIER and MASK sections

Data - CDF files

The CDF (Chip Description File) contains information about the layout of the chip. There is one for each chip type. So for an experiment you normally have only one. A CDF file allows you to

- Link between probes and probesets
- Identify which probes are PM and which are MM
- Identify control probes.

What is Bioconductor?

Bioconductor is “an open source and open development software project for the analysis and comprehension of genomic data”.

- Collection of packages implemented in the R language.
- Packages for
  - affy data
  - cDNA microarray data
  - multiple testing
  - Annotation
  - other .....

The *affy* Package

The component of Bioconductor designed for the analysis of Affymetrix GeneChip data.
Written by

- Rafael Irizarry
- Laurent Gautier
- Leslie Cope
- Ben Bolstad

I will discuss the Bioconductor 1.2 release version of the software.

Documentation for the *affy* Package

There is a reasonable amount of documentation about the *affy* package. An important source of information is the vignettes. These are report style documents describing different features. These can be accessed by typing *openVignette()*. Currently there are 5 vignettes for the *affy* package.

- **affy primer** the introductory document
- **affy: Built-in Processing Methods** documents pre-processing methods included in the package

Documentation for the *affy* Package

- **affy: Custom Processing Methods (HowTo)** describes how you might add your own pre-processing methods
- **affy: Import Methods (HowTo)*** describes how you might write routines to import data into the structures of the package.
- **affy: Automatic downloading of cdfenvs (HowTo)** Descibes how you can configure the package for downloading cdfenvs.

*Not as complete as one would like.

Installing and Loading the Package

- From within R install bioconductor

  source("http://www.bioconductor.org/getBioC.R")

  getBioC("all","release")

- This installs the latest release version of bioconductor
- load the *affy* library

  library(affy)

- This loads the *affy* library and its dependencies.

The *AffyBatch* Object

- main object for dealing with probe-level information
- stores probe intensity information for a collection of CEL files
- stores phenotypic information
- stores the name of the CDF file, MIAME information
- some useful methods

  pm() - get pm probe intensities

  mm() - get mm probe intensities

Reading in Data

Typically you have a set of CEL files that you want to read in. The easiest method for loading data is the *ReadAffy* command. Typically we would do something like

- Data <- ReadAffy()

which will attempt to load all the CEL files in the current directory (use *getwd()* and *setwd()*) into an *AffyBatch* object. If you have a large number of CEL files and a small amount of RAM this will typically take a very long time.

What About CDF Files?

We no longer need to worry about reading in CDF files. There are *cdfenv* packages which contain the processed CDF information needed by the package. These are ordinary R package. Using the current Bioconductor 1.2 release *cdfenvs* will be downloaded automatically when needed. If a processed *cdfenv* does not exist then it is possible to create one using the *makecdfenv* package.

Merging *AffyBatch* Objects

Sometimes new CEL files come along as an experiment progresses. Rather than having to re-run *Read.Affy()* with all of your files each time, it is possible to *merge* two *AffyBatch* objects together. eg

- *setwd("/a")*
- *Data1 <- ReadAffy()*
- *setwd("/b")*
- *Data2 <- ReadAffy()*
- *AllData <-merge(Data1,Data2)*

Subsetting *AffyBatch* Objects

Sometimes we want to leave certain cel files out of our analysis, or perform operations on a subset of the files. We can use standard indexing operations. So if *Data* is an *AffyBatch* then

- *Data[1:2]*
- *Data[c(2,4,6)]*
- *Data[-1]*

are all valid operations. Combining with the *merge* operation we could do something like *merge(Data1[-c(1,2,3)],Data2[c(4:6,9)])*.

Computing Expression Estimates

This is probably the most used functionality of the *affy* package. The function *expresso* is the general purpose function for computing expression estimates. *expresso* requires you to specify a processing method for each of four steps: background, normalization, pmcorrect and summary.
Another commonly used function is *rma* which computes RMA expression measure.

In general a method for computing expression measures should take an *AffyBatch* and output an *exprSet*.

The *exprSet* Object

- Object for storing expression measures, standard error estimates, phenotype data
- provided by the *Biobase* package. It is the basic object of *Bioconductor*
- the *AffyBatch* actually inherits many of its properties from the *exprSet*
- Some useful methods:

  exprs() - Extract expression estimates

  se.exprs() - Extract SE

RMA Expression Estimates

There are several options to compute RMA expression estimates

- *rma()* - The canonical implementation of the RMA algorithm. Implemented in C code with R wrapper so it is usually fastest.

```
eset <- rma(Data)
```

RMA Expression Estimates (more)

- *expresso()* - Using a particular set of choices for the preprocessing methods RMA expression estimates can be generated.

```
eset <-  expresso(Data,
 bgcorrect.method=''rma'',
 normalize.method=''quantiles'',
 pmcorrect.method=''pmonly'',
 summary.method=''medianpolish'')
```

RMA Expression Estimates (more)

- *justRMA()* This function avoids the *Affybatch* altogether. It reads the specified CEL files, parses the appropriate PM intensities, and then computes RMA (using the internal C routines of *rma*).

```
eset <- justRMA()
```

Getting Access to Probe-Level Data

There are several ways to get access to Probe level data. The accessor functions of the *AffyBatch* object *pm(Data)* and *mm(Data)* will return all the PM and MM probes on a chip as large matrices.
Sometimes it is more useful to get only the probes related to a single probeset (across all the arrays). It is less well know that you can use *pm* and *mm* to access individual probesets. For example

```
pm(Dilution,"107_at")
```

would return the PM intensities for probeset

```
107_at
```

Pre-Processing Probe Data

In general the background and normalization pre-processing methods will take a *AffyBatch* as an argument and return an *AffyBatch*. So it is very possible to preprocess and then examine probes for individual probesets as mentioned before.

The *AffyExtensions* Package

This package supplements and extends the functionality of the *affy* package. Available as an R package, but the bulk of the code is written in C. This ensures it is easy to use, but also utilizes the speed of compiled code.

The *PLMset* Object

This is the main object in the *AffyExtensions* package. It's purpose is to store information from a fitted probe level model. A *PLMset* object stores

- parameter estimates
- standard errors
- weights
- phenotype data
- cdf file name
- Common methods
  - *coefs()* - coefficents
  - *se()* - extract SE estimates
  - *image()* - pseudo chip images of the weights
  - *boxplot()* - boxplot of standardised SE

The *fitPLM* Command

This is the command for fitting a probe level model. It takes an *AffyBatch*, fits the specified model to each probeset, and produces a *PLMset*. A user may specify

- model to be fit (it is possible to fit more than just the probes + chips model of RMA)
- Preprocessing methods (beyond just the standard RMA methods)
- standard error method
- M-estimator to use (default is Huber, but a variety of others are now provided)

Some Examples

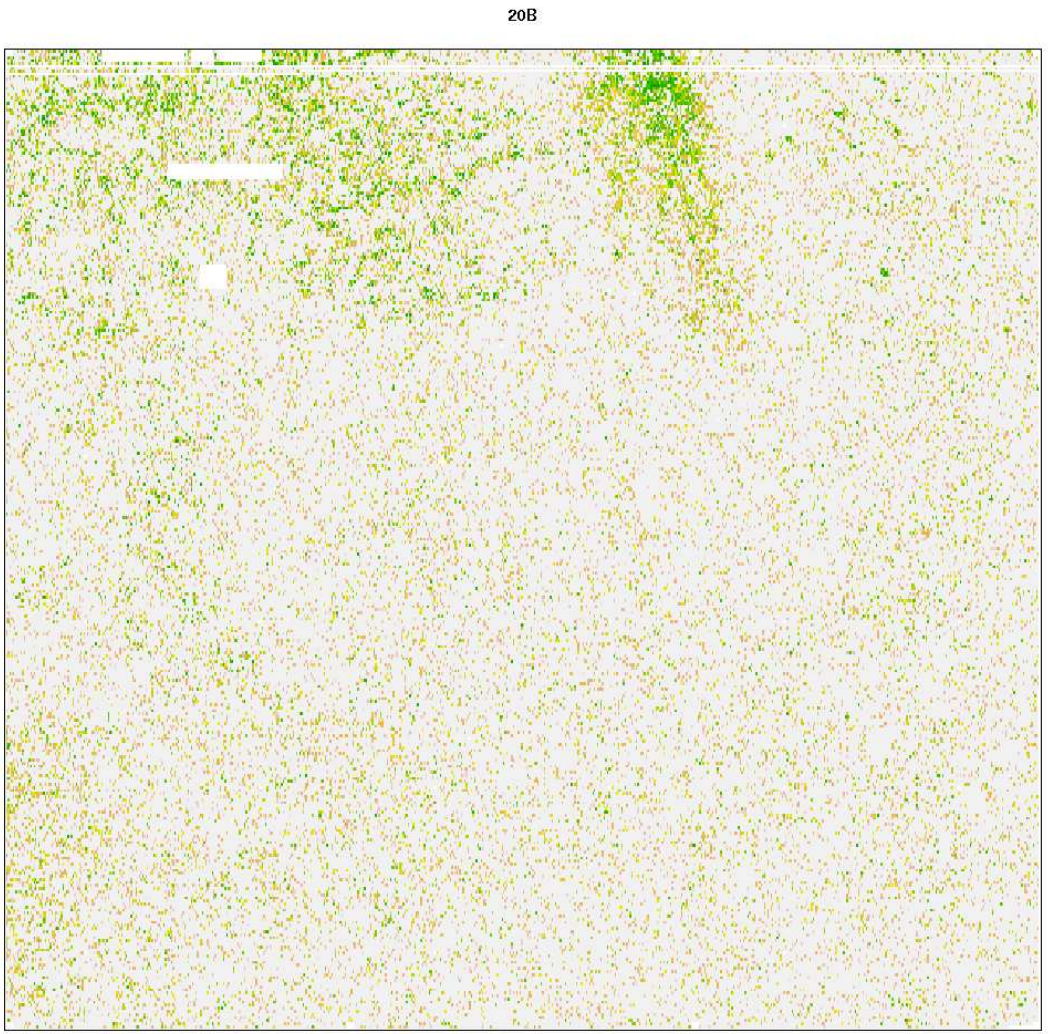
```
data(Dilution)

#Fit the default model
Pset <- fitPLM(Dilution)

#Fit with Tukey biweight M estimation
Pset <- fitPLM(Dilution,
          psi.type = ‘‘Tukey’’,
          psi.k=4.6851)

#Fit a model with Concentration and Scanner
Pset <- fitPLM(Dilution,
          model = PM ~ -1 + probes
                + liver + scanner)
```

Example Pseudo Chip Image



A Faster *ReadAffy*

Being unhappy with the performance of the *ReadAffy()* command I have rewritten the routine so that it reads all of the CEL files into one block of memory within C code, rather than reading CEL files one by one. This command is in the *AffyExtensions* package. To use it you should overload *read.affybatch* with *read.affybatch2*. So the command to type is

```
read.affybatch <- read.affybatch2
```

The hope is this will speed up the function greatly since the current routine is wasteful with memory (large objects are repeatedly copied) and when reading a larger number of cel files in it can get quite bogged down.

Timing results

Using the *system.time* facility we compare the routines performance for the two routines. The test chips were HGU95A.

Number of chips	Current (secs)	New (secs)
5	8.17	7.31
10	24.31	15.77
15	43.11	22.89
25	96.35	40.46
50	523.72	82.02
75	6306.10	188.97
98	?????	346.04

Test Machine

Component	Specification
Processor	AMD Athlon XP 2500+ (Barton)
RAM	1 GB
Swap Space	6 GB
OS	Linux (Redhat 9.0)
Kernel	2.4.21-rc7-ac1
R	1.7.1
affy	1.2.27
AffyExtensions	0.5-6

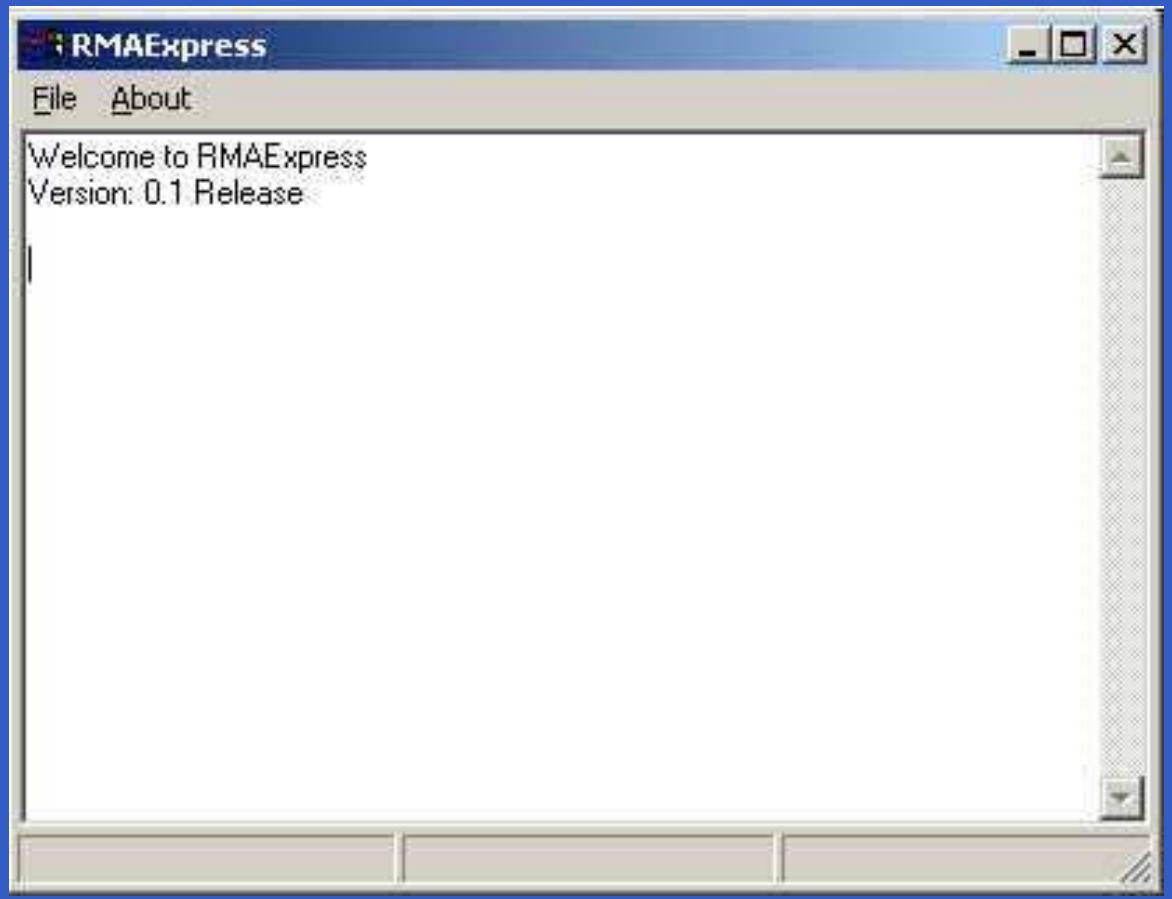
Timing conclusions

So we see the original routines propensity to copy large chucks of memory repeatedly forces the machine to swap heavily for an extended period of time making large datasets all but impossible to load. The new routine still does some unnecessary copying (I believe this is in the instantiation of the *AffyBatch* object) but does it only once saving a lot of running time.
We will seek to have the new function moved into the base *affy* package when it has been tested to be stable enough.

RMAExpress

- A simple program with a GUI interface aimed at Windows users.
- Implemented in C++. It has no dependencies on R.
- Generates RMA expression values.
- Requires text CEL and CDF files.

RMAExpress - A screenshot



Links

- Bioconductor http://www.bioconductor.org/
- AffyExtensions

  http://www.stat.berkeley.edu/users/bolstad/AffyExtensions/AffyExtensions.html
- RMAExpress

  http://www.stat.berkeley.edu/users/bolstad/RMAExpress/RMAExpress.html