aroma.affymetrix: A generic framework in R for analyzing small to very large Affymetrix data sets in bounded memory

Henrik Bengtsson[*], Ken Simpson[†], James Bullard[‡], Kasper Hansen[‡]

February 18, 2008

Abstract

Summary: We have developed a cross-platform open-source framework for analyzing Affymetrix data sets consisting of 1 to 1,000s of arrays. By working directly with CDF and CEL files (standard Affymetrix file formats) most chip types are automatically supported, e.g. expression, SNP, and exon arrays. The package provides methods for low-level analysis such as background correction of different kinds, allelic cross-talk calibration, quantile and affine normalization, PCR fragment-length and GC-content normalization, probe-level summarization such as robust log-additive and multiplicative modeling, as well as a set of methods for high-level analysis, e.g. chromosomal segmentation and alternative splicing. Results can be exported to dynamical HTML reports for easy navigation of a large set of arrays both offline and online. All algorithms have been optimized to run in bounded memory (as low as 500MB of RAM) by either redesigning the algorithms or by processing data in chunks. Transformed data and parameter estimates are stored on file in standard file formats, which in turn minimizes the memory overhead, but also makes them immediately accessible to other software. Moreover, storing intermediate results in persistent memory makes computational expensive analyses more robust against system failures and allows for quick resumes. In addition to making common algorithms readily available, this package was designed to allow for quicker development of novel models and incorporation of existing ones, such as Bioconductor methods, and be prepared for future chip types.

Availability: Software, documentation, examples and a user forum are available at *http://www.braju.com/R/aroma.affymetrix/*.

Contact: *hb@stat.berkeley.edu*

1 Introduction

Not only does the density of Affymetrix microarrays grow fast, but also the number of arrays per experiment. This steady growth in amount of data being produced is likely to continue for years ahead and is already causing problem for bioinformaticians and statisticians, because most available software, especially academically available software used for development of new models, does not handle today's large data sets. Moreover,

[*]Department of Statistics, University of California, Berkeley, USA.
[†]Walter & Eliza Hall Institute of Medical Research, Parkville, Australia.
[‡]Department of Biostatistics, University of California, Berkeley, USA.

when new chip types are released, it is not uncommon that there is a substantial latency before the researcher can start working with the new chip type, which in turn pushes the delivery of the final statistical methods further into the future.

To overcome these problems, we have developed *aroma.affymetrix* which can analyze most chip types with minimal setup efforts as well as virtually any number of arrays on not only large-memory systems but also systems with modest memory available.

The outline of this report is as follows: In Section 2 we briefly explain how we designed *aroma.affymetrix* to handle everything from small to very large data set. In Section 3, we provide a brief comparison to alternative academic frameworks for analyzing Affymetrix microarray data. In Section 4 the strict directory and filename structure used by the package is explained. In Sections 5-7, a minimal example is used to show how to use the package for low-level as well as high-level analysis and to illustrate how data is stored persistently on the file system. The report is concluded in Section 8.

2 Bounded-memory algorithms

Several pre-processing methods are based on single-array models, meaning their algorithms are typically designed to have bounded-memory properties. However, for other methods, such as the multi-array probe-level summarization methods described in Li and Wong (2001) and Irizarry *et al.* (2003), the nature of the underlying models often result in algorithms with a memory overhead that grows with the number of arrays as well as the number of units (genes, exons, SNPs, loci etc.) being processed. Fortunately, the above models are such that they are conditionally independent of unit, meaning they can be fitted unit by unit in bounded memory without having to turn to approximate algorithms. In order to optimize the processing time, *aroma.affymetrix* fits such unit-specific models in chunks, where only a subset of the units are modelled in each chunk. The amount of data loaded in each chunk can be scaled so that it utilizes an optimal fraction of the existing memory. In all cases, output data is stored on file (using standard file formats) for immediate access to downstream methods but also external software.

In addition, there exist multi-array methods for which the immediate naive algorithm is not bounded in memory, but that can be redesigned to run in bounded memory without loss. One example of this is the rank-based quantile normalization which in its original setup (Irizarry *et al.*, 2003) is non-bounded in memory, but by using a two-pass procedure (Bengtsson *et al.*, 2008) only holds data corresponding to two arrays in memory at any time.

For a summary of currently existing methods in *aroma.affymetrix*, see Table 1. Because all methods are bounded in memory, this means that literally any number of arrays can be processed by these methods. The only limitation is available disk space. Depending on chip type, most methods can in practice be run with as little as 0.5GB-1.5GB RAM. The largest data set known to have been processed by *aroma.affymetrix* to date consists of more than 5,000 HG-U133 expression arrays. The memory overhead for this was in the order of 0.5GB-1.0GB. As another example for the bounded-memory property, copy-number analysis of all 270 HapMap samples, measured on the latest GenomeWideSNP_6 chip type, can be done using approximately 1.5GB of RAM.

Method	*References*	*Multiarray model?*	*Bounded in memory*
Pre-summarization calibration & normalization			
Optical background correction	Wu *et al.* (2004)	single	yes
GCRMA background correction	Wu *et al.* (2004)	single	yes
RMA background correction	Irizarry *et al.* (2003)	single	yes
Rank-based quantile normalization*	Bolstad *et al.* (2003)	multi	yes
Function-based quantile normalization	Bengtsson *et al.* (2008)	multi	yes
Allelic crosstalk calibration	Bengtsson *et al.* (2008)	single	yes
Probe-level summarization			
Averaging model	(to be published)	single	yes
Log-additive model*	Irizarry *et al.* (2003)	multi	yes
Multiplicative model*	Li and Wong (2001)	multi	yes
Affine (multiplicative) model	Bengtsson *et al.* (2004)	multi	yes
Post-summarization calibration & normalization			
PCR fragment-length normalization	Bengtsson *et al.* (2008)	multi	yes
GC-content normalization	Bengtsson *et al.* (2008)	multi	yes
Equivalent fragment class normalization	(to be published)	multi	yes
Copy-number segmentation			
(Fast) CBS segmentation	Venkatraman and Olshen (2007)	single	yes
GLAD segmentation	Hupé *et al.* (2004)	single	yes
Alternative splicing			
FIRMA model	Purdom *et al.* (2008)	multi	yes

Table 1: List of low-level and high-level methods together with their memory properties as implemented in *aroma.affymetrix*. Methods that are adapted from other *R* packages are implemented such that the difference in output compared with the original implementation is minimal, if any. This is done by utilizing the original implementations as far as possible. *) Algorithm redesigned to be bounded in memory, i.e. to run with a constant memory overhead, but are otherwise giving the same results.

3 Brief comparison to other software

	aroma.affymetrix	*Affymetrix packages in Bioconductor*	*dChip*
Chip types normalized	All	All	Expression, SNP, exon
Chip types summarized	All with a CDF	All with existing CDF annotation packages	Expression, SNP, exon
Bounded-memory design	Yes	No, with some exceptions	Yes
Maximum number of arrays	Unlimited	Limited	Unlimited*
Persistent memory	Yes	No	Yes
Data handling	File system	Memory	File system
Read & writes CEL files	Yes	Read only, with some exceptions	Yes**
Third-party extensions	Yes	Yes	No
Operating system	Linux, Windows, OSX	Linux, Windows, OSX	Windows (Linux***)
License	LGPL v2	Artistic License v2, LGPL v2, ...	Freeware
Programming language	R	R	C++
Open source	Yes	Yes	On request
Active user forum	Yes	Yes	Yes

Table 2: Comparison with other freely available software that implement similar functionalities as *aroma.affymetrix*. *) Because dChip provides bounded-memory algorithms, theoretically there should be no upper limit in the number of arrays it can process. However, there seems to be a limit hardwired to 2000 arrays (*http://www.dchip.org/*). **) dChip writes CEL files in the ASCII format only, and for some chip types data is spatially rotated before being exported. ***) There is also an initiative to port dChip to Linux, but we have not investigated that project further.

The *aroma.affymetrix* framework was created in order to provide a user-friendly and memory-efficient framework that allows statisticians and bioinformaticians to setup and implement new statistical models (for Affymetrix analysis) with minimal effort, as well as end-users to use such models. This is also one of the main goals of the Bioconductor project (Gentleman *et al.*, 2004), but we note that there has only been little emphasize on providing bounded-memory algorithms. The Bioconductor project consists of a steady growing repository of packages covering a variety of fields and it is hard to generalize across packages and developers. There exist many Bioconductor packages for analyzing Affymetrix data, e.g. *affy*, *affycoretools*, *exonmap*, *affyPLM* and *oligo* to mention a few. For simplicitly we choose to do a brief comparison by putting such packages under the same umbrella. Another popular application for Affymetrix low-level analysis is the *dChip* software (*http://www.dchip.org/*; Li and Wong (2001)). Because it is freely available and widely used, we choose to compare the features of *aroma.affymetrix* to that software.

For a brief overview of the properties of *aroma.affymetrix*, Bioconductor packages for

Affymetrix analysis, and *dChip*, see Table 2.

4 Strict directory structure

The package enforces a strict directory structure assumed to be located in the current directory, making absolute pathnames obsolete. Because no absolute paths need to (should not) be specified, the same script can immediately be executed on a different platform with a different file system, simplifying collaboration but also allow for batch processing across hosts sharing the same file system. Furthermore, by using a standardized directory structure additional validation can be done by the software which further decrease the risk for incorrect analysis due to misplaced annotation and data files. We also hope that it will simplify troubleshooting both for the single user but also between collaborators.

Replication of directories and files existing elsewhere can be avoided by using soft links (Unix) or Windows Shortcut links (Windows). As explained next, a minimal setup consists of one directory tree for annotation data and one for raw data. Processed results are stored in similar directory structures as illustrated further in Section 5.

4.1 Annotation data

The *annotationData* directory tree holds all data that is not specific to a data set, e.g. CDF files. For further structuring, this directory contains a subdirectory named *chipTypes*, which in turn has one subdirectory for each chip type that contains annotation files from Affymetrix and other application, as illustrated below:

```
annotationData/
  chipTypes/
    Mapping250K_Nsp/
      Mapping250K_Nsp.CDF  Mapping250K_Nsp_annot.csv
    Mapping250K_Sty/
      Mapping250K_Sty.CDF  Mapping250K_Sty_annot.csv
```

With a strict directory structure for annotation data, there exist no ambigiouity to where to put files and less decisions to be made by the user.

4.2 Raw data

The *rawData* directory tree holds all data files specific to particular experiments. The data sets are structured in subdirectories, which in turn consists of subdirectories named exactly as the chip types hybridized. CEL files go into the latter, as illustrated below:

```
rawData/
  HapMap270,CEU/
    Mapping250K_Nsp/
      NA06985,B5,3005533.CEL  ...  NA12892,H5,4000092.CEL
    Mapping250K_Sty/
      NA06985,B5,3005533.CEL  ...  NA12892,H5,4000092.CEL
```

By enforcing that the subdirectories should have names corresponding to chip types, there is less room for mistakes and it is possible for *aroma.affymetrix* to further validate the correctness, which in turn increases the overall credibility of any analysis.

4.3 Format of file and directory name

File and directory names are interpreted such that for instance CEL files for the same sample on multiple chip types can be tupled, but also such that user-interpreted name tags can be used. File and directory names are parsed by the following grammar:

```
<filename> := <name>(,<tag>)*.<extension>
<dirname>  := <name>(,<tag>)*
```

For instance, from the above raw data directory, *aroma.affymetrix* knows that for data set 'HapMap270' (with tag 'CEU'), sample 'NA06985' has been hybridized to both Nsp and Sty. In addition to this, the above design makes it possible to locate data sets without having to specify paths to files and directories. For example,

```
cs <- AffymetrixCelSet$byName("HapMap270,CEU", chipType="Mapping250K_Nsp")
```

sets up the Nsp CEL set (without loading the actual data).

5 Low-level analysis

In this section will use a minimal example to show how data is analyzed in *aroma.affymetrix*. As shown in Table 1, there are many different pre-processing methods and here we will focus on two of the most common and generic ones namely quantile normalization and the log-additive model.

Given the setup outlined in previous section, rank-based quantile normalization (Bolstad *et al.*, 2003) can be done as:

```
qn <- QuantileNormalization(cs)
csN <- process(qn)
```

Note that the normalization is not performed until `process()` is called. The normalized data is stored as CEL files in:

```
probeData/
  HapMap270,CEU,QN/
    Mapping250K_Nsp/
      NA06985,B5,3005533.CEL  ...  NA12892,H5,4000092.CEL
```

By default, the above method adds a 'QN' tag to indicate that the outputted CEL files have been quantile normalized. Since the output is stored as standard binary CEL files, these can directly be used by other Bioconductor package, *dChip*, and all other software reading CEL files. Moreover, several methods exist for importing, into similar directory structures, data that has been exported by external applications such as *CNAG*, *CNAT*, and *dChip*, which makes it easy to also utilize other preprocessing software in the *aroma.affymetrix* pipeline.

Continuing, in order to fit a log-additive probe-level model (PLM) (Irizarry *et al.*, 2003), we first set up the model and then fit it as follows:

```
plm <- RmaPlm(csN)
fit(plm)
ces <- getChipEffectSet(plm)
```

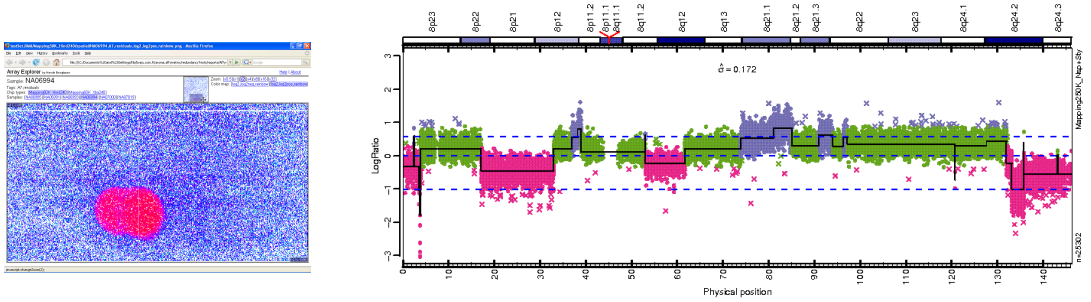

Figure 1: *Left*: The ArrayExplorer displaying falsely colored PLM residuals in the lower region of an HapMap 100K Hind240 array. *Right*: The main figure panel of a ChromosomeExplorer displaying raw copy numbers together with GLAD segmentation estimates on chromosome 8 in a tumor 500K sample. Using these "explorers", it is possible to zoom in on a spatial plot or along the chromosome, scroll, browse samples, chromosomes, chip types, color maps etc.

the estimated "chip effect" set, which contains the summarized data, is outputted under *plmData/HapMap270,CEU,QN,RMA/Mapping250K_Nsp/*, where the data set tag 'RMA' was append by the `RmaPlm` method.

Next, depending on chip type, summarized data may be normalized further. For options, see Table 1. In order to load a the data into memory for further processing, say by other packages, there exist a variety of support methods. For example, to extract the summarized probe signals across all arrays for a small set of units, do:

```
theta <- extractDataFrame(ces, units=c(1:100,800:812))
```

6 Dynamic reports

In addition to the above low-level methods, the *aroma.affymetrix* package can also generate dynamic HTML reports for easy navigator of data and model fits. For instance,

```
rs <- calculateResidualSet(plm)
ae <- ArrayExplorer(rs)
process(ae)
display(ae)
```

outputs images of PLM residuals and displays them in a web browser. The user interface is such that the residuals in regions of interested can quickly be zoomed into and browsed across arrays. See right panel of Figure 1 for an example.

7 High-level analysis

The *aroma.affymetrix* package also implements a small set of high-level analysis methods. Due to the field of research of the main developer, these methods are mainly for copy number analysis but there are also methods for alternative splicing.

As an example, below is a complete set of commands for normalizing Affymetrix SNP chip data, estimating raw copy numbers, identifying copy-number aberrations, and then displaying the raw copy numbers with identified regions as a dynamic HTML report in a web browser:

```
cs <- AffymetrixCelSet$byName("HapMap270,CEU", chipType="GenomeWideSNP_6")
acc <- AllelicCrosstalkCalibration(cs)
csC <- process(acc)
plm <- RmaCnPlm(csC, combineAlleles=TRUE, mergeStrands=TRUE)
ces <- fit(plm)
fln <- FragmentLengthNormalization(ces)
cesN <- process(fln)
seg <- CbsModel(ces)
ce <- ChromosomeExplorer(seg)
process(ce)
```

For an example of the output, see right panel of Figure 1. For details of each of the pre-processing steps in the above script, please see Bengtsson *et al.* (2008) as well as the online vignettes.

8 Conclusions

By storing intermediate results on file, not only is it possible to process 1,000s (and likely also 10,000s) of arrays, but we are also secured against system failures. When restarting an interrupted analysis script, possibly on a different machine on the same file system, it will quickly skip already processed data and proceed to the latest step.

Aroma.affymetrix is robust, memory efficient, and can easily be extended with new methods and algorithms. Currently, there are several background, pre-processing, probe-level modeling, post-processing, as well as some alternative splicing and copy-number methods implemented. All algorithms have been designed to run in bounded memory, which means that virtually all analyses can be conducted even on limited systems such as a regular notebook, something which is not possible with current Bioconductor implementations. If more powerful systems are available, *aroma.affymetrix* can scale its analysis accordingly.

Acknowledgement

Thanks to B. Bolstad (Affymetrix) for packages *affyPLM* and *preprocessCore*, which is heavily used in several of the summarization models. HB was supported by grants from the Wenner-Gren Foundation (Stockholm), the American-Scandinavian Foundation (Stockholm & New York), and the Solander Foundation (Lund University).

References

Bengtsson, H., Jönsson, G., and Vallon-Christersson, J. (2004). Calibration and assessment of channel-specific biases in microarray data with extended dynamical range. *BMC Bioinfo.*, **5**, 177.

Bengtsson, H., Irizarry, R., Carvalho, B., and Speed, T. P. (2008). Estimation and assessment of raw copy numbers at the single locus level. *Bioinformatics*.

Bolstad, B., Irizarry, R., Åstrand, M., and Speed, T. (2003). A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, **19**(2), 185–93.

Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Li, F. L. C., Maechler, M., Rossini, A. J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, J. Y. H., and Zhang, J. (2004). Bioconductor: Open software development for computational biology and bioinformatics. *Genome Bio.*, **5**, R80.

Hupé, P., Stransky, N., Thiery, J.-P., FranoisRadvanyi, and Barillot, E. (2004). Analysis of array CGH data: from signal ratio to gain and loss of DNA regions. *Bioinformatics*, **20**(18), 3413–3422.

Irizarry, R. A., Bolstad, B. M., Collin, F., Cope, L. M., Hobbs, B., and Speed, T. P. (2003). Summaries of Affymetrix GeneChip probe level data. *Nucleic Acids Res.*, **31**(4), e15.

Li, C. and Wong, W. (2001). Model-based analysis of oligonucleotide arrays: expression index computation and outlier detection. *Proc. Natl. Acad. Sci. USA*, **98**(1), 31–6. dchip.

Purdom, E., Simpson, K., Robinson, M., Conboy, J., Lapuk, A., and Speed, T. (2008). FIRMA: a method for detection of alternative splicing from exon array data. Submitted.

Venkatraman, E. S. and Olshen, A. B. (2007). A faster circular binary segmentation algorithm for the analysis of array CGH data. *Bioinformatics*.

Wu, Z., Irizarry, R., Gentleman, R., Murillo, F. M., and Spencer, F. (2004). A model based background adjustment for oligonucleotide expression arrays. *J. Am. Stat. Assoc.*, **99**(1001), 909–917. available at http://ideas.repec.org/p/bep/jhubio/1001.html.