Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *http://www.jstatsoft.org/*

Using the R Package crlmm for Genotyping and Copy Number Estimation

Robert B Scharpf
Johns Hopkins University

Rafael A Irizarry
Johns Hopkins University

Matthew E Ritchie
Walter+Eliza Hall Institute of Medical Research

Benilton Carvalho
University of Cambridge

Ingo Ruczinski
Johns Hopkins University

Abstract

Genotyping platforms such as Affymetrix can be used to assess genotype-phenotype as well as copy number-phenotype associations at millions of markers. While genotyping algorithms are largely concordant when assessed on HapMap samples, tools to assess copy number changes are more variable and often discordant. One explanation for the discordance is that copy number estimates are susceptible to systematic differences between groups of samples that were processed at different times or by different labs. Analysis algorithms that do not adjust for batch effects are prone to spurious measures of association. The R package **crlmm** implements a multilevel model that adjusts for batch effects and provides allele-specific estimates of copy number. This paper illustrates a workflow for the estimation of allele-specific copy number and integration of the marker-level estimates with complimentary Bioconductor software for inferring regions of copy number gain or loss. All analyses are performed in the statistical environment R. A compendium for reproducing the analysis is available from the author's website (http://www.biostat.jhsph.edu/~rscharpf/crlmmCompendium/index.html).

Keywords: copy number, batch effects, robust, multilevel model, high-throughput, oligonucleotide array.

1. Introduction

Duplications and deletions spanning kilobases of the genome contribute to a substantial pro-

portion of the genetic variation between individuals. Copy number variants (CNV) account for a greater proportion of differences in terms of sequence composition between two individuals than single nucleotide polymorphisms (SNPs) (Zhang *et al.* 2009). CNV can arise through a number of mechanisms during meiosis and mitosis and are well known to be implicated in cancer through deletions that disrupt tumor suppressor genes or the amplification of oncogenes. Copy number alterations have also been implicated in several genomic disorders, including complex diseases such as schizophrenia and autism (Karayiorgou *et al.* 2010; Pinto *et al.* 2010).

Current estimates regarding the frequency and size of segmental duplications and deletions in the human genome are largely based on high-throughput arrays that quantitate copy number on a genomic scale. Two such technologies are array comparative genomic hybridization (aCGH) and *genotyping* platforms such as the Affymetrix oligonucleotide arrays and the Illumina BeadArrays. While each of these platforms rely on the hybridization of probes to sample preparations containing target DNA sequence, differences exist in the size of the probes, the number of probes per target sequence, and whether the hybridization is competitive. Unlike aCGH, genotyping arrays can be used to identify copy-neutral regions of homozygosity that, while common in apparently normal individuals, can suggest rare genetic events such as uniparental isodisomy (UPD). UPD has been implicated in heritable diseases such as Prader-Willi syndrome (Altug-Teber *et al.* 2005). While the resolution is potentially much greater in genotyping arrays due to the shorter probe length, shorter probe lengths tend to result in more probe-to-probe variability with respect to cross-hybridization to the alternative allele, nonspecific binding, and differences in basepair composition. Reliable inference of copy number gain or loss at a single 25 - 100 basepair locus is not currently possible, and statistical methods that smooth the locus-level estimates as a function of the physical position in the genome are needed.

Despite robust-to-outlier approaches for normalization, we have observed systematic differences in the copy number between groups of samples that can be perfectly predicted by the timestamp on the CEL files. We refer to such systematic difference in copy number between groups of samples as batch effects. That larger studies tend to have more substantial batch effects than smaller studies is consistent with our conjecture that the nonstatic nature of experimental reagents and laboratory conditions contribute over time to batch effects. Irrespective of etiology, we have found that the scan date of the array and chemistry plate are useful surrogates for batch (Scharpf *et al.* 2011). With an appropriate experimental design that involves randomization of samples to chemistry plate, batch effects are a nuisance variable that can be successfully modeled and removed.

Existing analytic strategies for identifying alterations in copy number have largely adopted a one- or two-step approach. In the one-step approach, assessments of CNV are made from the raw intensities using the joint distribution across samples. For instance, Zhang *et al.* (2009) developed a Correlation Matrix Diagonal Segmentation (CMDS) that identifies recurrent alterations in a population. While we have not formally evaluated the impact of batch effects using this approach, it is important to note that the differences in raw intensities between groups of samples, whether driven by biological causes or by technological artifacts such as batch effects, are similar in terms of their effects on the data. A safe strategy when adopting such an approach would be to filter loci associated with experimental factors such as chemistry plate or scan date.

In contrast to the one-step approach, two-step approaches generally derive estimates of copy

number and uncertainty at each marker, followed by smoothing of the marker-level estimates at the second stage. The motivation for the two-step approach is that the marker-level estimates are too imprecise to provide reliable copy number estimates. However, marker-specific estimates can be useful for at least two reasons. First, single-locus estimates are typically derived from the joint distribution of intensities across samples and, through inspection of the joint distribution, batch effects can be modeled and removed. Secondly, plots of the marker-level estimates can be useful for assessing copy number mosaicism. Mosaicism occurs when mixtures of cell populations with different mutations give rise to noninteger copy number estimates. For instance, many tumors are comprised of a mixture of cell populations representing different levels of tumor evolution. The choice of appropriate statistical methods for smoothing at the second stage can therefore be informed by visualizations of the marker-level estimates. In particular, hidden Markov models (HMMs) (Fridlyand *et al.* 2004; Colella *et al.* 2007; Wang *et al.* 2007; Scharpf *et al.* 2008) are generally more appropriate for germline diseases in which latent, integer copy number states are reasonable. By contrast, segmentation algorithms such as circular binary segmentation (Olshen *et al.* 2004; Venkatraman and Olshen 2007) may be more appropriate for diseases such as cancer. While segmentation algorithms estimate segment means, HMMs provide direct inference about the latent copy number states of interest and can be used to identify copy-neutral regions of homozygosity.

This paper describes software for the first of a two-stage approach for identifying CNV in high-throughput genotyping arrays. All software was written using the statistical language R (R Development Core Team 2011). We illustrate our approach on 1258 HapMap samples that were assayed on the Affymetrix 6.0 platform (Consortium *et al.* 2010). Section 2 gives a brief overview of the the statistical methods for preprocessing, genotyping, and copy number estimation. Section 3 describes current data structures used by the **crlmm** package for organizing and annotating large datasets, as well as the organization of the compendium package for reproducing the figures in this manuscript. Section 4 describes the 2-step approach for identifying copy number alterations, highlighting the type of exploratory data analysis made possible in the **crlmm** package. Closing remarks are provided in Section 5.

2. Methods

This section describes the steps for processing the raw fluorescence intensities from scanned arrays, genotyping the polymorphic markers, and deriving bivariate normal prediction regions for allele-specific copy number. The bivariate normal prediction regions have several possible useful applications with respect to copy number estimation, including a simple translation to estimates of raw copy number at each marker.

2.1. Preprocessing

Preprocessing refers to normalization of the raw fluorescence intensities to remove array-to-array variability and the summarization of the normalized intensities of replicate probes. The **crlmm** package adapts the robust multichip average (RMA), originally described for gene expression microarrays (Irizarry *et al.* 2003), to genotyping platforms. We refer to this algorithm as SNP-RMA (Carvalho *et al.* 2007). Recent platforms for Affymetrix and Illumina include probes for polymorphic loci as well as probes for nonpolymorphic regions. At polymorphic loci, the raw intensities for each allele are quantile normalized (Bolstad *et al.* 2003) to a tar-

get reference distribution obtained from the HapMap phase 2 samples (Carvalho *et al.* 2007). The Affymetrix 6.0 platform contains 3 or 4 identical probes for each allele. The normalized intensities for a set of identical probes are summarized by the median. For nonpolymorphic loci, only one probe per loci is available and the intensities are quantile normalized without a subsequent summarization step. Additional details regarding the preprocessing of Affymetrix CEL files and Illumina IDAT files are described elsewhere (Carvalho *et al.* 2007; Ritchie *et al.* 2009).

2.2. Genotyping

Following the normalization and summarization of intensities at polymorphic loci, we apply the corrected robust linear mixture model (CRLMM) algorithm to genotype SNPs. This algorithm extends previous algorithms for genotyping, namely RLMM (Rabbee and Speed 2006) and BRLMM (Affymetrix 2006). A key difference of our approach and our predecessors is the use of HapMap samples to train our algorithm. The CRLMM algorithm originally described in Carvalho *et al.* (2007) has been adapted to accommodate changes in the technology of the more recent Affymetrix 5.0 and 6.0 platforms. Specifically, the probes for each locus are identical and lie on the same strand for the 5.0 and 6.0 platforms, simplifying the estimation procedure. In addition, the implementation of CRLMM in the **crlmm** package does not provide a correction for fragment-length as the improvement to model fit has, in our experience, not justified the additional computation. A recent comparison paper describes genotyping algorithms for Illumina's Infinium arrays (**?**).

The CRLMM algorithm provides genotype calls and quality scores for all polymorphic markers through a hierarchical model described in detail elsewhere (Carvalho *et al.* 2010). One critical aspect of our the procedure is to formulate the genotype classification problem in the space of the log-ratio of the observed intensities I for the A and B alleles versus the overall strength of the A and B intensities. More precisely, the y and x axes in a M versus S scatter plot are defined by $\log_2(I_A/I_B)$ and $\log_2(\sqrt{I_A \times I_B})$, respectively. Our previous work has demonstrated that the M are more robust to batch effects than the bivariate distribution of $\log A$ and $\log B$. For example, see Supplementary Figure 1 in Scharpf *et al.* (2011). We also note that the separation of the M values for the diallelic genotypes g, $g \in \{AA, AB, BB\}$, is smaller for SNPs with very low or very high values of S. For example, Figure 7B of Carvalho *et al.* (2007) depicts the relationship of M and S for one sample. Taken together, these observations motivated the development of a hierarchical model to account for systematic sources of variation from batch and intensity strength. Following the estimation procedure for model parameters outlined elsewhere (Carvalho *et al.* 2010), the inferred genotypes are based on the derivation of the posterior probability for the true genotype Z. Assuming that there are no batch effects, a simplification of the posterior probability for SNP i in sample k is given by

$$\mathsf{P}(Z_{ik} = g|M_{ik} = m) = \frac{\mathsf{P}(Z_{ik=g})h_{M_{ik}|Z_{ik}=g}(m)}{\sum_{g'} \mathsf{P}(Z_{ik=g'})h_{M_{ik}|Z_{ik}=g'}(m)} \text{ for } g' \in \{AA, AB, BB\}, \quad (1)$$

where $h_{m_{ik}|Z_{ik}}(m)$ represents a normal density. See equations (4) and (6) of Carvalho *et al.* (2010) for the derivation of the mean and variance for the normal density. The genotype call, $\hat{Z}_{ik}$, and confidence score, q_{ik}, for SNP i in sample k are given by

$$\hat{Z}_{ik} = \arg\max_g \mathsf{P}(Z_{ik} = g|M_{ik} = m) \text{ and} \quad (2)$$

$$q_{ik} = \max_g \mathsf{P}(Z_{ik} = g|M_{ik} = m), \quad (3)$$

respectively. Our implementation maps the scores q_{ik} from the $[0, 1]$ interval to integers using the relationship

$$M(q_{ik}) = \text{round}(-1000 * \log 2(q_{ik})),$$

as it allows efficient storage with minimum loss.

In summary, the CRLMM algorithm estimates genotypes through a hierarchical model for the log ratios of A:B intensities that accounts for the dependency on intensity strength, batch effects, and the uncertainty of parameters estimated from the training step. For each platform design supported by our software, we provide one annotation package that contains parameters estimated from the training data for every SNP-genotype combination.

2.3. Bivariate normal prediction regions for integer copy number

In large studies, batch effects become evident as the strength of the A and B intensities is sensitive to changes to laboratory conditions, reagents, or personnel that change over time. Estimation of absolute copy number is a more difficult enterprise than genotyping in part because batch effects and true differences in copy number are similar in terms of their effects on the data. While quantile normalization is an effective means for removing array to array variation and provides additional robustness to outliers in individual samples, such normalization procedures are insufficient for removing batch effects. However, algorithms that assign biallelic genotypes to samples based on the ratio of log intensities, as implemented in the **crlmm** algorithm, are robust to batch effects. We utilize CRLMM's robustness to batch effects to guide the estimation of parameters in a multilevel model for copy number.

This section briefly describes the algorithm for estimating batch-specific bivariate normal prediction regions of integer copy number. Typically, the 96-well chemistry plate is a useful surrogate for batch as the samples from a given plate are often processed at similar times. As indicated above, the resulting prediction regions can be used to (i) compute a posterior mean copy number at each marker, (ii) incorporated directly into a hidden Markov model to infer regions of copy number gain and loss, or (iii) used to derive an estimate of *raw* copy number. The estimation procedure extends probe-level models for estimating gene expression (see Wu and Irizarry (2005)) and an early version of an algorithm for estimating copy number described by Wang *et al.* (2008). As of this writing, the algorithm requires the genotypes of the experimental dataset and does not use any training data, such as HapMap, as priors. As a consequence, the current implementation requires a minimum of 10 samples per batch for estimating parameters for copy number, with larger batch sizes (e.g, 90 or more samples) preferred.

At each SNP, we (i) calculate robust estimates of the mean and variance for each of the diallelic genotypes, (ii) shrink the empirical estimates to the within-batch averages estimated from a large number of SNPs, (iii) impute the location and variance of unobserved genotypes, and (iv) fit a linear model to the within-genotype cluster medians. More formally, we propose the following theoretical model for the observed intensity I at SNP i, batch j, sample k, and allele l:

$$\begin{aligned} \left[\ I_{ijkl}\ \right] &= \left[\left(\ \text{Optical}_{ijl} + \text{Nonspecific}_{ijl}\ \right) \times \left(\ \delta_{ijkl}\ \right)\right] + \left[\ \text{Specific}_{ijl} \times \varepsilon_{ijkl}\ \right] \\ &\equiv \left[\ \nu_{ijl} \times \delta_{ijkl}\ \right] + \left[\ \phi_{ijl} c_{ijkl} \times \varepsilon_{ijkl}\ \right] \text{ for } l \in \{A, B\}, \end{aligned} \tag{4}$$

where the errors δ and ε account for array to array variation within a batch and are assumed to be approximately log-normal. Fluorescence arising from nonspecific hybridization and optical background are collectively parameterized by ν. To estimate the ν and ϕ in model (4), we assume that the c_l are known from the diallelic genotype calls. For instance, c_A takes the value 0, 1, or 2 for genotypes BB, AB, and AA, respectively. Next, we fit a regression line to estimates of the median intensity for each genotype stratum using weighted least squares (equation (7) of Scharpf *et al.* (2011)). The intercept and slope of the regression line correspond to our estimates of ν_A and ϕ_A in model (4), respectively. We repeat the procedure for allele B. As in Wang *et al.* (2008), we assume that the joint distribution of the log intensities conditional on the allelic copy number is approximately bivariate normal:

$$\left[\begin{array}{l} \log_2(I_{ijkA}) \\ \log_2(I_{ijkB}) \end{array} \middle| \begin{array}{l} C_{ijkA} = c_A \\ C_{ijkB} = c_B \end{array} \right] \sim \mathrm{N}\left(\left[\begin{array}{l} \log_2(\nu_{ijA} + c_A\phi_{ijA}) \\ \log_2(\nu_{ijB} + c_B\phi_{ijB}) \end{array} \right], \ \mathbf{\Sigma}_{ij} \right) \tag{5}$$

We obtain initial estimates of the covariance Σ empirically, and subsequently shrink the the covariance to improve estimates for genotypes with few observations (Scharpf *et al.* 2011). The means of the bivariate normal in equation 5 are obtained by plugging in estimates of ν and ϕ from the linear model. Section 4 provides code for plotting the predictions regions as well as the *raw* copy number. The raw copy number is the sum of the allele-specific estimates, $\hat{c}_A + \hat{c}_B$, obtained through the following relationship:

$$\hat{c}_{ijkl} = \max\left\{ \frac{1}{\hat{\phi}_{ijl}} \left(I_{ijkl} - \hat{\nu}_{ijl} \right), \ 0 \right\} \text{ for } l \in \{A, B\}. \tag{6}$$

For nonpolymorphic markers and monomorphic SNPs, we follow a similar procedure using SNPs with complete data to predict the unobserved genotypes. For example, the unobserved 'A' and 'null' genotypes for monomorphic AA SNPs and nonpolymorphic loci are imputed from a regression model using SNPs with all three diallelic genotypes observed as explanatory variables. The linear model is fit to the imputed genotype-cluster medians and variances using weighted least squares as described previously (Scharpf *et al.* 2011).

3. Implementation

This section provides a brief overview of this compendium and the data structure adopted by **crlmm** for binding information on the samples, markers, and experiment in a single object.

Compendium: This document is written using `Sweave` (Leisch 2003) and is included in the **crlmmCompendium** R package. The **crlmmCompendium** package can be downloaded from `http://www.biostat.jhsph.edu/~rscharpf/crlmmCompendium`. The raw CEL files used in this analysis are publicly available (`http://hapmap.ncbi.nlm.nih.gov/downloads/raw_data/hapmap3_affy6.0`). Manageable subsets of the markers and samples from the processed data are included with the **crlmmCompendium** package for the purpose of reproducing the figures and illustrating key features of the software. The website for the compendium places code extracted from this `Sweave` document for each of the figures alongside thumbnail versions of the figures. Reproducing the complete analysis described in this `Sweave` file requires two

additional steps. First, one would need to obtain the CEL files for the HapMap phase 3 data and verify that any additional R packages beyond those that are required for installing the compendium are available. See Section 6 for the complete R session information from our analysis. Secondly, the following code chunk specifying the path to the CEL files and the directory to store results should be edited as appropriate.

```
R> outdir <- "<PATH_TO_CEL_FILES>"
```

In addition to the above tasks, we suggest caching long computations such that repeated `Sweave` calls can be loaded from disk. We used the R package **cacheSweave** (http://cran.r-project.org/web/packages/cacheSweave/index.html) for this purpose and alert the user to blocks of code in Section 4 that may be useful for caching (Peng and Eckel 2009). Subsequent `Sweave`'s calls of this file are fast and can be performed in an interactive session as cached computations are lazy loaded into the global environment.

crlmm: The R package **crlmm** is available from Bioconductor (Gentleman *et al.* 2004). New releases of **crlmm** are available twice per year. The **crlmm** package utilizes the S4 class *CNSet* container for encapsulating the normalized intensities and various other aspects of the experiment and samples. The *CNSet* class extends the basic *eSet* container defined in **Biobase**. As with other *eSet* extensions, phenotypic data on the samples, annotation for the probes, and meta-data on the experiment are also included in the container. Through inheritance, all of the general methods defined for the *eSet* class are available for the *CNSet* class. The *CNSet* is specialized for copy number estimation as follows. Elements of the `assayData` slot include the normalized intensities for the A and B alleles, the CRLMM genotype calls, and the CRLMM confidence scores. Data and meta-data for the samples and probes are stored in the `phenoData` and `featureData` slots, respectively. The class defines two additional slots important for copy number estimation: `batch` and `batchStatistics`. The `batch` slot contains a character vector that describes the batch in which the CEL files were processed. The character vector must be the same length as the number of samples. The `batchStatistics` slot is similar to the `assayData` slot, but as the name implies the elements in the environment are SNP- and batch-specific statistical summaries needed for copy number estimation, including robust estimates of the mean and variance for each genotype cluster and parameters from the linear model in equation (4). Each element in the `batchStatistics` environment has dimension $I \times J$, where I is the total number of markers (SNPs and nonpolymorphic probes) and J is the total number of batches. Several examples for accessing the batch statistics are illustrated in Section 4. Objects of the *CNSet* class are typically instantiated by the `genotype`. However, new objects can also be derived by the "[" method that subsets rows (markers) and columns (samples).

`Parallelization and large data support:` Several of the algorithms described in the previous section have been written to allow parallelization of computation across multiple processors or nodes. For instance, the SNP-RMA and CRLMM algorithms, as well as the algorithm for copy number estimation allow for parallel computing. The infrastructure for parallel computing is handled by the R package **snow** available from CRAN (http://cran.r-project.org).

We have made several adaptations to reduce the memory footprint of key functions for processing large datasets. The reduction is primarily made possible by the use of data structures and protocols provided by the **ff** package (http://cran.r-project.org/web/packages/ff/

`index.html`) for storing objects on disk rather than in memory. Algorithms such as SNP-RMA and CRLMM require only subsets of the samples or markers, respectively. As a consequence, data can be read into memory, processed, and summaries written to file without excessive memory requirements. We manage the location of files on disk and the size of the data chunks (subsets of rows and/or columns) through three utility functions provided as part of the **oligoClasses** package: `ldPath`, `ocProbesets`, and `ocSamples`. Documentation for these functions is available in the **oligoClasses** package and illustrated in Section 4. Currently, all of the elements in the `assayData` slot and `batchStatistics` slot are **ff** objects, as opposed to ordinary matrices. While the use of **ff** objects has many advantages, operations on these objects require more careful attention to avoid accidentally calling too much data from disk and swamping the available memory. In general, this can be achieved by specify both the i and j arguments to the "[" method. However, one should be careful in the order of operations for accessing data from a *CNSet* object. For example, the normalized A allele intensities for the first 5 SNPs could be obtained by either of the following commands:

```
R> as.matrix(A(cnSet)[1:5, ])
R> as.matrix(A(cnSet[1:5, ]))
```

While the results from both commands would be identical, the first command is the preferred approach as the I/O for the second command can be substantially greater. In particular, the second command subsets every element in the `assayData` and `batchStatistics` slot prior to extracting the normalized A allele intensities. Note also that we have wrapped both calls by the function `as.matrix`. This is important when dealing with **ff** objects as the object returned by assessors for `assayData` and `batchStatistics` can be either of class *data.frame* or *matrix*, depending on the size of the data set. Explicit coercion to class matrix avoids bugs that could arise, for instance, by a method that behaves differently for *data.frame* and *matrix* objects.

Known limitations: A limitation of the current design is that the `assayData` elements must all have the same dimension. However, the nonpolymorphic markers only interrogate one allele and we do not estimate a genotype at these markers. Implicitly, the assay data elements for allele B, the genotype calls, and the genotype confidence scores for nonpolymorphic markers are larger than required and store many `NA`s. For example, the matrix for genotype calls in the Affymetrix 6.0 platform are almost twice as large as needed. The advantage of equally-sized assay data elements is that (i) information on the features can be represented in a single location that is bound to the assay data and feature annotation, (ii) subsetting objects of the class does not require any special handling, and (iii) fewer bugs due to the simplicity of the design and the inheritance of well-tested methods that are defined for the *eSet* class. We prefer the added reliability of the current structure to the potential improvements in efficiency, particularly since the implementation does not generally effect the memory requirements as the **ff** package stores the data on disk.

Currently, copy number estimation is not available for samples in which the batch size is fewer than 10. Unlike the $\log_2(A/B)$ ratio, the strength of A and B intensities is sensitive to batch making training with data such as HapMap more difficult. The practical consequence is that **crlmm** does not currently estimate the parameters in the linear model for batches with fewer than 10 samples, and may provide noisy estimates of copy number for batches with fewer than 50 samples. Future versions of **crlmm** may include solutions for small data scenarios.

Finally, several files for storing the data will be created by utilities in the **ff** package. If these files are later moved to a different location or removed, the accessors for data in the *CNSet* object will no longer work. Users should either use the `clone` utility in the **ff** package for relocating files on disk, or be prepared to rerun the genotyping and copy number estimation steps. As in any statistical analysis using R, saving the exact session information can be useful for reproducing a previous analysis.

4. Results

This section describes a 2-step approach for identifying regions of copy number alterations. In the first step, raw copy number estimates at each marker are obtained using the **crlmm** package. In the second step, the raw copy number estimates are smoothed using a hidden Markov model implemented in the **VanillaICE** package and by circular binary segmentation implemented in the **DNAcopy** package. The visualizations provided in this section use **lattice** graphics for effective multi-panel displays (Sarkar 2008).

4.1. Fitting the multilevel model

We begin our analysis of the HapMap data by loading the compendium and various dependencies for our analysis of the HapMap data.

```
R> library("ff")
R> library("Biobase")
R> library("genefilter")
R> library("IRanges")
R> library("MASS")
R> library("VanillaICE")
R> library("crlmmCompendium")
```

As the result of the above commands, the **ff** package is in the search path and support for large datasets is automatically enabled. We set the path to store objects on disk and fine-tune the size of the data chunks used by the CRLMM algorithm using the three utility functions mentioned in the previous section.

```
R> ldPath(outdir)
R> ocProbesets(50e3)
R> ocSamples(200)
```

Finally, the **cacheSweave** package is loaded for caching output from long code chunks and a directory for storing the cached computations is declared.

```
R> library("cacheSweave")
R> setCacheDir(outdir)
```

We complete the set-up for our analysis of the HapMap samples by specifying the names of the CEL files and defining a surrogate for batch. As indicated previously, a useful surrogate for batch is the scan date of the array or the chemistry plate. For the HapMap phase 3 data, the chemistry plate is the first 5 letters of the CEL filename. We extract the plate names from the filenames in the following code.

```
R> filenames <- list.celfiles(pathToCels, full.names=TRUE, pattern=".CEL")
R> plate <- substr(basename(filenames), 1,5)
```

The steps for preprocessing and quantile-normalizing Affymetrix CEL files in **crlmm** are wrapped in the function `genotype`. (Users that only require the genotype calls and do not intend to estimate copy number should use the `crlmm` function instead.) The `genotype` function is a wrapper for several important steps. First, the function initializes an object of class *CNSet*, verifying the validity of the data passed to the `batch` argument of this function. Secondly, the function reads the raw intensities from the CEL files and normalizes the intensities in a memory efficient manner *via* the SNP-RMA algorithm (Bolstad *et al.* 2003; Carvalho *et al.* 2007). The nonpolymorphic markers are also quantile-normalized to a target reference distribution estimated from HapMap. Finally, we call the CRLMM algorithm to genotype and estimate genotype confidence scores at SNPs. As the preprocessing and genotyping is computationally intensive, we set `cache=TRUE` in the declaration of the following code chunk.

```
R> cnSet <- genotype(filenames=filenames, cdfName="genomewidesnp6", batch=plate)
```

The object returned by the `genotype` function, assigned to the name `cnSet` in the above command, is an instance of the S4 class *CNSet*. In addition to specifying batch as an argument to `genotype`, one may optionally specify the gender. If gender is not specified as in the above example, the gender is imputed. Typically, the imputed gender should be accurate as a large number of markers are available to estimate gender. However, in cancer samples or diseases with chromosome X or Y aneuploidy, the gender calls may be incorrect or ambiguous. Otherwise, one reason to allow the imputation is as a check for possible inconsistencies in the supplied documentation. The `show` method for class *CNSet* provides a concise summary of its contents.

```
R> show(cnSet)
```

```
CNSet (assayData/batchStatistics elements: ffdf)
CNSet (storageMode: lockedEnvironment)
assayData: 1852426 features, 1258 samples
  element names: alleleA, alleleB, call, callProbability
protocolData
  rowNames:
    CHEAP_p_HapMapP3Redo2_GenomeWideSNP_6_A04_235570.CEL
    CHEAP_p_HapMapP3Redo2_GenomeWideSNP_6_A07_235576.CEL ...
    TESLA_p_CEU_Trio_GenomeWideSNP_6_A03_223714.CEL (1258
    total)
  varLabels: ScanDate
  varMetadata: labelDescription
phenoData
  sampleNames:
    CHEAP_p_HapMapP3Redo2_GenomeWideSNP_6_A04_235570.CEL
    CHEAP_p_HapMapP3Redo2_GenomeWideSNP_6_A07_235576.CEL ...
    TESLA_p_CEU_Trio_GenomeWideSNP_6_A03_223714.CEL (1258
    total)
```

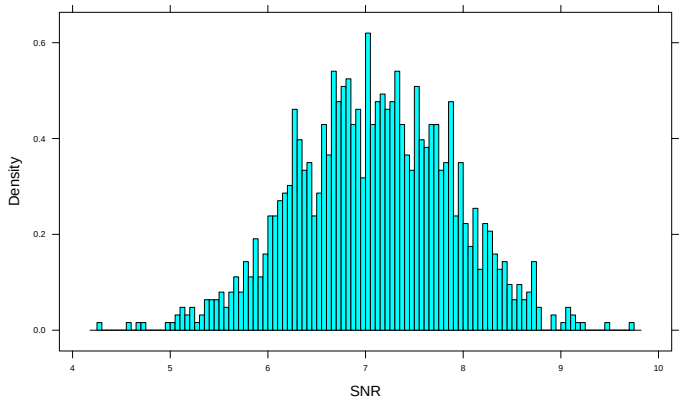

Figure 1: Signal to noise ratio for the HapMap phase 3 data. Signal to noise ratios below 5 can indicate problems with data quality.

```
  varLabels: SKW SNR gender
  varMetadata: labelDescription
featureData
  featureNames: SNP_A-2131660 SNP_A-1967418 ... CN_929945
    (1852426 total)
  fvarLabels: isSnp position chromosome
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation: genomewidesnp6
batch:    CHEAP 8, CORER 8, CUPID 77, DINGO 73, EPODE 89, ...
batchStatistics:  29  elements,  1852426  features,  19  batches
```

Following preprocessing and genotyping, the SNR (see Section 2) can be a useful statistic for quality control. In some instances, it may be preferable to remove samples with low SNR from downstream analyses. The SNR is stored in the `phenoData` slot of the `cnSet` object and can be accessed and plotted as follows. See Figure 6 for a histogram of the SNR values in the HapMap phase 3 study.

```
R> open(cnSet$SNR)

[1] FALSE

R> SNR <- cnSet$SNR[]
R> close(cnSet$SNR)

[1] TRUE

R> (snrfig <- histogram(~SNR, breaks=100))
```

By default, the sample names for the `cnSet`, accessible by `sampleNames(cnSet)` are the CEL filenames. The **crlmmCompendium** contains a mapping from the CEL filenames to the

more familiar HapMap identifiers. The following code changes the sample labels from the filename to the HapMap identifiers. Note that we generally suggest using the filenames for the sample names, but have adopted the more concise HapMap names in this analysis to improve readability of the resulting output.

```
R> cnSet$celFiles <- sampleNames(cnSet)
R> sampleNames(cnSet) <- getHapMapIds(cnSet)
R> sampleNames(cnSet)[1:5]
```

The metadata on the samples and features can be listed with the `varLabels` and `fvarLabels`, respectively.

```
R> varLabels(cnSet)

[1] "SKW"      "SNR"      "gender"   "celFiles"

R> fvarLabels(cnSet)

[1] "isSnp"      "position"   "chromosome"
```

Recall that the assay data elements in the `cnSet` object contain pointers to large objects on disk. One can list all of the **ff** files created during the initialization of the container as in the following code chunk. Once created, these files should not be moved or deleted.

```
R> list.files(ldPath(), pattern="\\.ff$")[1:2]

[1] "baf_chr1056e332943b61.ff" "baf_chr1156e377005dd8.ff"
```

The underlying data structures are intended to be handled seamlessly through the provided interface in **crlmm**. For instance, in the following code chunk we open file connections to the **ff** objects and access the quantile normalized intensities for the first 5 markers and the first 6 samples for allele A.

```
R> invisible(open(cnSet))
R> as.matrix(A(cnSet)[1:5, 1:6])

             NA12045 NA20881 NA18873 NA19652 NA19679 NA18625
SNP_A-2131660    3263    5087    4655    3225    3368    4736
SNP_A-1967418     180     233     188     271     228     267
SNP_A-1969580     577     724     782     752     486     751
SNP_A-4263484    2100    1287    1658     462    2205     336
SNP_A-1978185    1571    1369    2116    1810    1861    1292
```

The above query is not instantaneous as these items pull data from the `ff` files on disk to active memory. The bracket operator without arguments, as in `[,]`, would pull data from all markers and all samples from disk to active memory, defeating the purpose of using the **ff** package.

In the analysis of genomewide association data, it is often useful to visualize the genotype clusters for loci of interest. All that is required for such a visualization is the platform-specific identifier for the SNP of interest. In the example below, we plot the genotype clusters for SNP_A-2131660. The object `genotypeSet` that contains the data for this SNP is available in the **crlmmCompendium** package and was generated from the following commands.

```
R> invisible(open(cnSet))
R> ##genotypeSet <- cnSet[match("SNP_A-4247386", featureNames(cnSet)), ]
R> genotypeSet <- cnSet[match("SNP_A-2131660", featureNames(cnSet)), ]
R> invisible(close(cnSet))
```

The R function `prePredictPanel` in the **crlmmCompendium** package extracts the normalized intensities, the genotype calls, and the confidence scores for the genotypes and stores the results in an object of class *data.frame*. The resulting *data.frame* object will be useful for creating trellis displays with the **lattice** package.

```
R> df <- prePredictPanel(genotypeSet)
```

We will construct 3 scatterplots of this SNP using colors to annotate different aspects of the data. The following code selects a color for the plotting symbols that indicates the CRLMM genotype call.

```
R> fill1 <- brewer.pal(3, "Set1")[df$gt]
```

CRLMM provides a genotype call for *all* SNPs (no missing values) and a confidence score for the called genotype (equation (2)). If one wishes to exclude SNPs with low confidence scores, visualizations of the confidence scores in the context of the scatterplot can be useful. In the following code, we select a grey scale for the confidence score with darker shades of grey indicating less confidence ranging to which (confidence = 1). As the confidence scores for this SNP were all high (≥ 0.89), we selected a scale such that scores near 0.89 are shaded dark.

```
R> gt.conf <- df$gt.conf
R> min.conf <- min(gt.conf)
R> max.conf <- max(gt.conf)
R> sc <- (gt.conf - min.conf)/(max.conf-min.conf)
R> fill2 <- sapply(sc, grey)
```

Finally, we choose a subset of samples to highlight the batch effects frequently observed in large studies in which the samples are processed over an extended period of calendar time. As mentioned previously, the scan date of the array or the chemistry plate are often useful surrogates for batch effects. The scan dates of the array are stored in the `protocolData` slot of the *CNSet* object and can be extracted using the $ operator:

```
R> dt <- strftime(protocolData(genotypeSet)$ScanDate, "%Y-%m-%d", usetz=FALSE)
R> range(dt)
```

```
[1] "2007-04-03" "2008-04-19"
```

We expect that the normalized intensities for samples processed near the beginning of the study may be systematically different from samples processed near the end of the study. As we are using plate as a surrogate for batch, we select two plates in which the samples were processed at very different times.

```
R> dt.batch <- split(dt, batch(genotypeSet))
R> sapply(dt.batch, range)
```

```
     CHEAP        CORER        CUPID        DINGO
[1,] "2008-04-17" "2008-03-19" "2007-09-17" "2007-08-30"
[2,] "2008-04-19" "2008-04-02" "2007-09-18" "2007-09-14"
     EPODE        GIGAS        HOMOS        HUFFS
[1,] "2007-08-30" "2007-04-06" "2007-09-14" "2008-01-19"
[2,] "2007-09-13" "2007-05-25" "2007-09-17" "2008-01-22"
     HUSKS        IONIC        LOVED        NIGHS
[1,] "2008-01-31" "2007-09-14" "2007-09-14" "2007-10-27"
[2,] "2008-02-01" "2007-09-17" "2007-09-15" "2007-11-06"
     PICUL        POSIT        SAKES        SCALE
[1,] "2007-08-30" "2007-08-30" "2007-08-30" "2007-04-03"
[2,] "2007-09-13" "2007-09-13" "2007-12-18" "2007-04-04"
     SHELF        SLOTH        TESLA
[1,] "2007-04-06" "2008-01-19" "2008-02-06"
[2,] "2007-04-09" "2008-01-22" "2008-02-06"
```

Note that the samples on the SCALE plate were processed at the beginning of April of 2007, whereas samples on the SLOTH plate were processed in January of 2008. We select different colors for SCALE and SLOTH, and use white for the remaining samples.

```
R> batch.scale <- which(batch(genotypeSet)=="SCALE")
R> batch.sloth <- which(batch(genotypeSet)=="SLOTH")
R> plate.cols <- brewer.pal(8, "Accent")[c(3, 8)]
R> fill3 <- rep("white", nrow(df))
R> fill3[batch.scale] <- plate.cols[1]
R> fill3[batch.sloth] <- plate.cols[2]
```

Next we replicate the *data.frame* object 3 times, attaching a different set of fill colors for each replicate. The factor `colorby` will be used as a conditioning variable in the lattice graphic such that a separate panel is created for each factor.

```
R> df2 <- rbind(df, df, df)
R> df2$fill <- c(fill1, fill2, fill3)
R> colorby <- c("genotype", "confidence score", "plate")
R> df2$colorby <- factor(rep(colorby, each=nrow(df)), levels=colorby, ordered=TRUE)
```

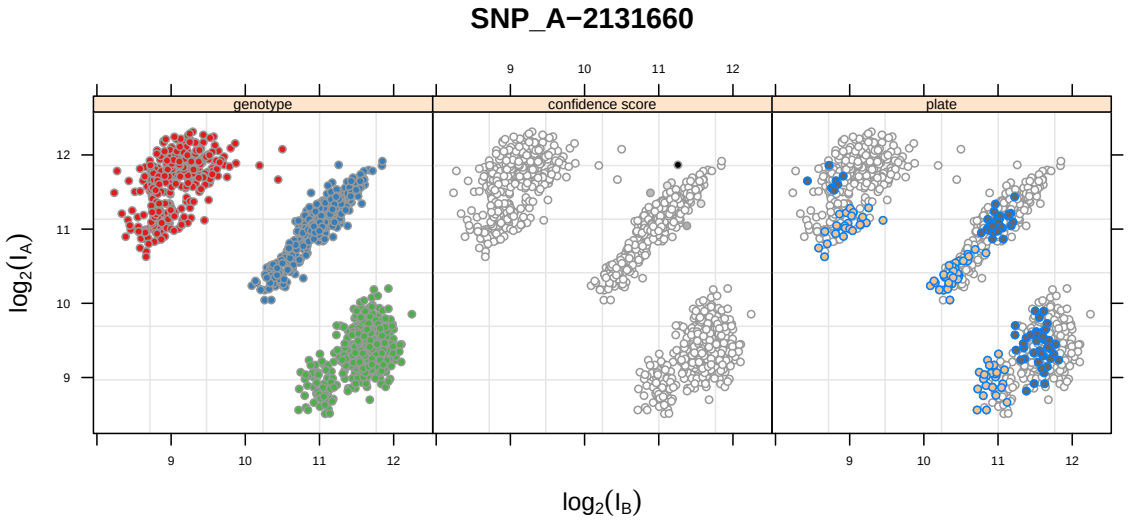

Figure 2: Scatterplots of the normalized log A versus log B intensities for one SNP. The panel labels indicate whether the plotting symbols are colored by genotype (left), the CRLMM genotype confidence score (middle), or chemistry plate (right). The CRLMM confidence scores were all high, ranging from 0.89 (dark grey) to 1 (white). The HapMap phase 3 samples were processed over a time interval of approximately 1 year. The two highlighted in the right panel had scan dates that were processed approximately 8 months apart.

The following call to the `xyplot` creates the desired multi-panel display in Figure 2. Note that the plate- (batch-) effect is in the A+B direction and that the genotypes are robust to the batch-effect. As described in Section 2, our copy number analysis will use chemistry plate as a surrogate for batch and the robust-to-batch CRLMM genotypes to train the linear model.

```
R> (ABfig <- xyplot(A~B|colorby, df2,
                panel=function(x, y, col, fill, plate.cols, ..., subscripts){
                        panel.grid(h=5,v=5)
                        panel.xyplot(x,y, col="grey60", fill=fill[subscripts], ...)
                        if(panel.number() == 3){
                                ## such that these plates are plotted last
                                lpoints(x[batch.scale], y[batch.scale], fill=plate.cols[1],
                                lpoints(x[batch.sloth], y[batch.sloth], fill=plate.cols[2],
                        }
                },
                 aspect="iso",
                 fill=df2$fill, col=df2$col, cex=0.6, pch=21, plate.cols=plate.cols,
                 xlab=expression(log[2](I[B])),
                 ylab=expression(log[2](I[A])), main=featureNames(genotypeSet), layout=c(3,
                 par.strip.text=list(lines=0.9, cex=0.6)))
```

Alternatives. Alternatives to the the R package **crlmm** for genotyping Affymetrix arrays include **BRLMM** (Rabbee and Speed 2006; Affymetrix 2006), **Birdseed** (Korn *et al.* 2008; Mc-

Carroll *et al.* 2008), **SNPiPer-HD** (Hua *et al.* 2007), **CHIAMO** (Wellcome Trust Case Control Consortium 2007), and the Affymetrix Genotyping Console (**GTC**) software. (**GTC** uses Birdseed to genotype Affymetrix 6.0 arrays and BRLMM to genotype Affymetrix 5.0 arrays.) An alternative to quantile-normalization for preprocessing Affymetrix arrays is implemented in the R package **aroma.affymetrix** (Bengtsson *et al.* 2008).

4.2. Marker-level copy number estimation

Following preprocessing and genotyping by the `genotype` function, we call the R function `crlmmCopynumber` to estimate the parameters for copy number estimation outlined in Section 2. The `crlmmCopynumber` function requires an object of class `CNSet` and returns the value `TRUE` upon successful completion. Additional arguments to the `crlmmCopynumber` are available and are documented in the **crlmm** package. As with the `genotype` function, we set the `cache=TRUE` flag in the delimiter for the following code chunk.

```
R> invisible(open(cnSet))
R> cnSet.updated <- crlmmCopynumber(cnSet)
```

Note that the `crlmmCopynumber` returns `TRUE` upon successful completion and does not return an object of class *CNSet*. Rather, updates to elements of the `batchStatistics` slot in the `cnSet` object are written to disk using the **ff** interface and are not returned by the function.

Batch-specific summary statistics are computed as part of the copy number estimation step. Accessors defined in the **crlmm** package return these summary statistics as arrays. The following code chunk illustrates a few of the available accessors for batch-specific summary statistics, including the genotype frequencies (`Ns`), the median absolute deviation (across samples) of the normalized intensities (`mads`), and the median intensity (`medians`). The argument j is used to indicate the batch. In the example below, we extract the above summary statistics for the 3rd and 4th batch.

```
R> table(batch(cnSet))
R> Ns(cnSet, i=1:3, j=3:4)
R> mads(cnSet, i=1:3, j=3:4)[, "A", , ]
R> medians(cnSet, i=1:3, j=3:4)[, "A", , ]
```

The regression coefficients from model (4) for copy number are also stored in the `batchStatistics` slot. A useful means to inspect model fit is to plot the normalized intensities and overlay the fitted regression line. In the following code chunk, we instantiate a new `CNSet` object containing 16 randomly selected SNPs and all of the samples on the GIGAS chemistry plate.

```
R> invisible(open(cnSet))
R> invisible(open(cnSet$gender))
R> set.seed(123)
R> snp.index <- sample(which(isSnp(cnSet) == 1), 16, replace=FALSE)
R> sample.index <- which(batch(cnSet) == "GIGAS")
R> exampleData1 <- cnSet[snp.index, sample.index]
R> invisible(close(cnSet))
R> invisible(close(cnSet$gender))
```

The `exampleData1` object is provided in the **crlmmCompendium** package and can be loaded as follows.

```
R> if(!exists("exampleData1")) data(exampleData1)
```

Next we extract the normalized intensities for the A and B alleles, the genotype calls, and the estimated coefficients for the intercept (nuA) and slope (phA) for the A allele. Figure 3 illustrates the fit of the linear model to the A allele intensities for the SNPs in the `exampleData1` object.

```
R> a <- t(as.matrix(A(exampleData1)))
R> gt <- t(as.matrix(calls(exampleData1)))
R> nuA <- as.numeric(nu(exampleData1, "A"))
R> phA <- as.numeric(phi(exampleData1, "A"))
R> col <- brewer.pal(7, "Accent")[c(1, 4, 7)]
R> NN <- Ns(exampleData1, i=1:16, j=1)[, , 1]
R> fns <- featureNames(exampleData1)
R> snpId <- matrix(fns, nrow(a), ncol(a), byrow=TRUE)
R> snpId <- factor(snpId, levels=fns, ordered=TRUE)  ##IMPORTANT
R> ldat <- data.frame(A=as.integer(a),
                      gt=as.factor(c("AA", "AB", "BB")[as.integer(gt)]),
                      snp=snpId)
R> boxplotfig <- bwplot(A~gt|snp, ldat, cex=0.6, panel=lmPanel, nu=nuA,
                        ph=phA, fill="lightblue", Ns=NN,
                        par.strip.text=list(lines=0.9, cex=0.6),
                        ylab=expression(I[A]),
                        xlab=expression(I[B]), ltext.y=2500, label.cex=0.6)
```

Scatterplots of the log-transformed normalized intensities for the A and B alleles can be useful for visualizing the bivariate normal prediction regions for integer copy number (see equation (5)). Using the same set of randomly selected SNPs in the previous code chunk, we plot the prediction regions for copy numbers 0, 1, 2, 3, and 4. To set up this graphic, we use two functions provided with the **crlmmCompendium** package: `prePredictPanel` and `makeTransparent`. The `prePredictPanel` and `makeTransparent` functions are used to organize the genomic data into a *data.frame* object and to allow a partially transparent rendering of the prediction regions, respectively. Finally, we set up a legend for the figure using the **lattice** function `simpleKey` and create an object of class *trellis* with the function `xyplot`. The resulting trellis object is displayed in Figure 4

```
R> ldat <- prePredictPanel(exampleData1)
R> shades <- makeTransparent(brewer.pal(6, "BrBG"), alpha=0.6)[c(1,2,3,5,6)]
R> ##replace the middle color (white) with something else
R> mykey <- simpleKey(as.character(0:4), points=FALSE, rectangles=TRUE, col="black", space
R> mykey$rectangles[["col"]] <- shades
R> (bvnfig <- xyplot(A~B|snp, ldat, cex=0.3, panel=cnPredictionPanel, object=exampleData1,
                     x.axis="B", copynumber=0:4, line.col=shades, line.lwd=1.5,
                     shades=shades, ylab=expression(log[2](I[A])), xlab=expression(log[2](I[B])
```

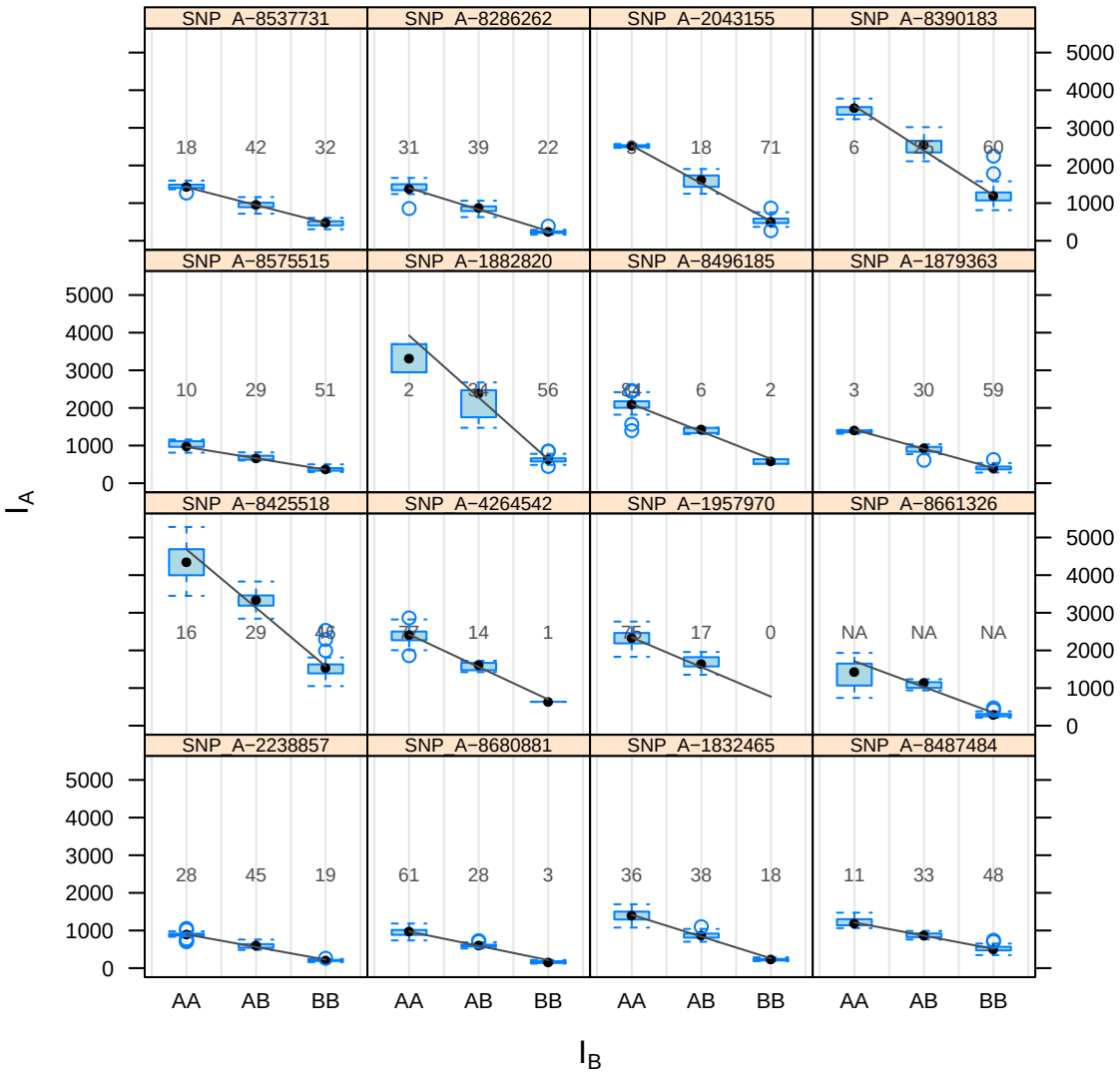

Figure 3: Each panel displays the intensities for the A allele for all samples on the GIGAS plate stratified by the genotype call. The linear model is fitted on the intensity scale (as opposed to the log-scale) with parameters for the intercept and slope that are SNP- and batch-specific. The straight line over-plotted is the estimated background and slope for the GIGAS plate. The numbers in each panel indicate the genotype frequencies.

```
par.strip.text=list(lines=0.9, cex=0.6),
 xlim=c(6,12.5), ylim=c(6,12.5),
key=mykey))
```

In most applications, the raw copy number estimates are intermediate values and passed directly to segmentation algorithms or hidden Markov models. Rather than estimate the raw copy number for all markers and samples and deal with I/O of data storage and access,

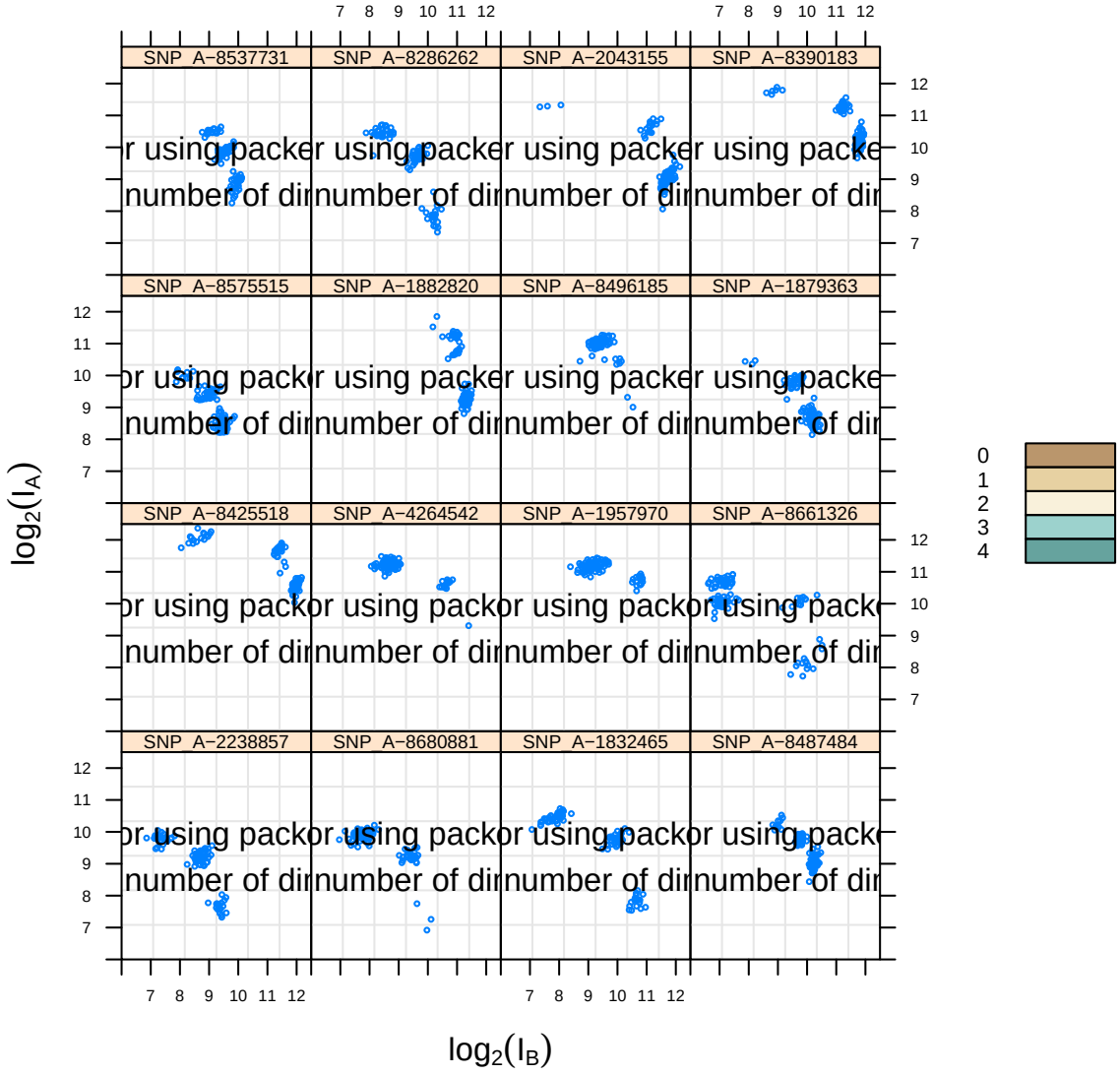

Figure 4: A scatter plot of the log 2 normalized intensities for 16 randomly selected SNPs. The bivariate normal prediction regions derived from the linear model are plotted for copy numbers 0 - 4.

our preference is to summarizes these estimates by the genomic intervals obtained by by a segmentation or hidden Markov model. In the following code, we create a *CNSet* object containing only the markers on chromosome 8 and the samples in batch SHELF. (For reasons discussed in the following section, we are particularly interested in the NA19007 sample.)

```
R> marker.index <- which(chromosome(cnSet) == 8)
R> sample.index <- match("NA19007", sampleNames(cnSet))
R> batch.index <- which(batch(cnSet) == batch(cnSet)[sample.index])
R> invisible(open(cnSet))
```

```
R> shelfSet <- cnSet[marker.index, batch.index]
R> shelfSet <- shelfSet[order(position(shelfSet)), ]
R> invisible(close(cnSet))
R> dup.index <- which(duplicated(position(shelfSet)))
R> if(length(dup.index) > 0) shelfSet <- shelfSet[-dup.index, ]
```

Using the relationship defined in equation (6), we obtain estimates of the total copy number for the `shelfSet` object using the `rawCopynumber` function. Specifically, this function computes $\hat{c}_A + \hat{c}_B$. The `robustSds` function provided in the **VanillaICE** package can be used to provide estimates of uncertainty.

```
R> tcn <- rawCopynumber(shelfSet, i=seq(length=nrow(shelfSet)), j=seq(length=ncol(shelfSet
R> sds <- robustSds(tcn)
```

Alternatives. An alternative to absolute allele-specific copy number estimation in **crlmm** is the R package **aroma.affymetrix** that provides estimates of copy number relative to a reference set (Bengtsson *et al.* 2008).

4.3. Downstream tools for inferring regions of copy number gain and loss

Marker-level estimates of copy number for Affymetrix and Illumina platforms are too noisy to reliably quantitate copy number at a single marker. Approaches that smooth the copy number estimates as a function of the physical position are useful for inferring regions of copy alterations and copy-neutral regions of homozygosity (ROH). This section illustrates how the marker-level estimates of copy number from **crlmm** can be passed to downstream segmentation and HMM algorithms. We illustrate our approach on chromosome 8 of HapMap sample NA19007 for which a large amplification on the p-arm has been previously identified (Redon *et al.* 2006).

For fitting a HMM or segmenting estimates of copy number, it is convenient to put estimates of copy number into a container for genotypes and copy number. The container `oligoSnpSet` defined in the **oligoClasses** package serves this purpose and can be instantiated directly from a *CNSet* object. The `redonSet` object created in the following code chunk is provided with the compendium.

```
R> sample.index <- match("NA19007", sampleNames(shelfSet))
R> j <- match("NA19007", sampleNames(shelfSet))
R> redonSet <- new("oligoSnpSet",
                   copyNumber=integerMatrix(tcn[, j, drop=FALSE], 100),
                   cnConfidence=integerMatrix(1/sds[, j, drop=FALSE], 100),
                   call=as.matrix(calls(shelfSet)[, j, drop=FALSE]),
                   callProbability=as.matrix(snpCallProbability(shelfSet)[, j, drop=FALSE]),
                   phenoData=phenoData(shelfSet)[j, ],
                   featureData=featureData(shelfSet))
```

Centering the copy number estimates by the median for the chromosome ensures that the baseline state is centered at 2. Note that for cancer samples, centering by the median copy number calculated across all autosomes would enable one to identify deletions or duplications that effect most of a chromosome.

```
R> copyNumber(redonSet) <- copyNumber(redonSet) - median(copyNumber(redonSet), na.rm=TRUE)
```

A hidden Markov model. The HMM implemented in the R package **VanillaICE** allows some flexibility for the data inputs and the definition of the hidden states. For example, the vignette included in the **VanillaICE** package documents how one can fit a HMM to copy number-only data (e.g, if genotypes were not available as in array comparative genomic hybridization), copy number and genotype data (SNP chips), and genotype-only data. Of course, the hidden states depend on the data type. For copy number and genotype data, one could have copy number alterations as well as copy-neutral regions of homozygosity as hidden states. For genotype-only data, the hidden states are region of homozygosity and normal. In the following code, we specify homozygous deletion, hemizygous deletion, normal, and amplification as the hidden states of interest. For each of the hidden states, the user must indicate the corresponding initial state probability (log-scale) and the probability of a homozygous genotype. The object returned by the `hmm.setup` function contains various parameters used for fitting the HMM as well as the estimated emission probabilities.

```
R> cnStates <- c(0, 1, 2, 2, 3, 4)
```

Next, we apply the Viterbi algorithm to estimate the optimal sequence of states (Viterbi 1967). The transition probabilities used in the HMM are a function of the distance between markers and can be scaled by the `TAUP` object to control the smoothness of the state path. In particular, larger values of `TAUP` make it more difficult to transition between states and provide a smoother state path. Future versions of the **VanillaICE** package may estimate `TAUP`.

```
R> fit.cn <- hmm(redonSet, TAUP=1e10, cnStates=cnStates, is.log=FALSE)
R> hmm.df <- as.data.frame(fit.cn)
R> print(hmm.df[, c(2:4,7, 10,11)])
```

```
      start       end    width      id state          LLR
1     31254   3612943  3581690 NA19007     3     0.000000
2   3613110   5780785  2167676 NA19007     6   743.373666
3   5782057  42436956 36654900 NA19007     3     0.000000
4  42441576  43824048  1382473 NA19007     4    11.327760
5  46847534  49102747  2255214 NA19007     6   -59.516253
6  49103034  50003194   900161 NA19007     4     5.745739
7  50004901  50127899   122999 NA19007     6    -8.365361
8  50128277  82961379 32833103 NA19007     3     0.000000
9  82962713  83623619   660907 NA19007     4    32.468397
10 83624332  84495551   871220 NA19007     6    -2.203806
11 84496173  85085400   589228 NA19007     4     4.845637
12 85085979  85459776   373798 NA19007     6   -15.338185
13 85464065  86557413  1093349 NA19007     4    40.785295
14 86835379 146298155 59462777 NA19007     5 -1903.306472
```

The evidence for the deletions and alternations is summarized by the log likelihood ratio (LLR) comparing the predicted amplification or deletion to the null model of no copy number alteration. The LLR can be a useful statistic for ranking copy number alterations.

```
R> hmm.df[hmm.df$state ==5, "LLR"]

[1] -1903.306
```

Circular binary segmentation. For somatic cell diseases such as cancer, DNA is collected from tissue that may contain a mixture of cell populations. For example, cells that represent different stages of cancer evolution. At any given locus, it is possible that a fraction of the cells have different integer copy numbers. As a result, the aggregate copy number measured by an array is not necessarily an integer and the concept of a genotype in a mixture of cell types is ambiguous. Unlike hidden Markov models that often assume an integer copy number state, segmentation algorithms identify genomic segments with constant copy number. Note that there is no implicit assumption that the copy number at any given locus is an integer. As a result, segmentation algorithms such as circular binary segmentation (Olshen *et al.* 2004; Venkatraman and Olshen 2007) are well-suited for the analysis of genomic data from cancers.

In this section, we fit the circular binary segmentation (CBS) algorithm to the copy number estimates from HapMap sample NA19007. The CBS algorithm is implemented in the R package **DNAcopy**. Following the vignette accompanying the **DNAcopy** package, we create an object of class *CNA* and use the `smooth.CNA` to smooth single point outliers.

```
R> CNA.object <- CNA(genomdat=copyNumber(redonSet)/100,
                     chrom=chromosome(redonSet),
                     maploc=position(redonSet),
                     data.type="logratio",
                     sampleid=sampleNames(redonSet))
R> smu.object <- smooth.CNA(CNA.object)
```

Next, we apply the CBS algorithm to the smoothed data using the function `segment` in the package **DNAcopy**. As the segmentation can be slow, we use the `cache=TRUE` in the declaration of the following code chunk.

```
R> cbs.segments <- segment(smu.object)
R> print(cbs.segments, showSegRows=TRUE)
```

The output from the `segment` function is a collection of genomic intervals annotated by the mean copy number and the number of markers in the segment. While the above example contains genomic ranges from the segmentation of a single subject's chromosome 8, a typical analysis may contain multiple subjects and chromosomes. The **IRanges** package available from Bioconductor provides an extensive infrastructure for manipulating and organizing genomic ranges, as well as efficient functions for common queries such as `findOverlaps` that operate on objects of the class. We therefore illustrate how one can extract the segmentation results and create an object of class *RangedData* defined in the **IRanges** package that may be useful in downstream applications. The functions `RangedData` and `IRanges` in the following code instantiate instances of the corresponding class. Additional details on these classes and functions are provided in the help files and vignettes accompanying the **IRanges** package available from Bioconductor.

```
R> cbs.out <- cbs.segments$output
R> cbs.segs1 <- RangedData(IRanges(cbs.out$loc.start, cbs.out$loc.end),
                           numMarkers=cbs.out$num.mark,
                           seg.mean=cbs.out$seg.mean,
                           chrom=8L)
```

The helper function `addCentromereBreaks` included in the **crlmmCompendium** is used to add breaks for the centromere. The copy number estimates and genotype calls are then collected into a simple *data.frame* that will be passed to the **lattice** function `xyplot` for visualizing the data.

```
R> cbs.segs1 <- addCentromereBreaks(cbs.segs1)
R> cn <- as.numeric(copyNumber(redonSet))
R> gt <- as.integer(as.matrix(calls(redonSet)))
R> df <- data.frame(cn=cn, gt=gt, position=position(redonSet)/1e6)
```

Finally, we select colors for distinguishing homozygous from heterozygous genotypes, and a second set of colors to indicate the hidden states inferred from the hidden Markov model.

```
R> genotype.cols <- c("lightblue", "green3", "lightblue")
R> states <- unique(as.integer(factor(fit.cn$state, levels=c(1, 3, 4, 5))))
R> shades <- brewer.pal(10, "PRGn")
R> shades <- shades[c(2,4,1,8)]
R> shades[3] <- "white"
R> shades <- makeTransparent(shades, alpha=0.6)
R> mykey <- simpleKey(c("hom-del", "hem-del", "normal", "duplicated")[states[order(states)
                      rectangles=TRUE, col="black", space="top", cex=0.7)
R> mykey$rectangles[["col"]] <- shades[states[order(states)]]
```

The panel function `cnPanel` provided with the **crlmmCompendium** is used to plot the the copy number and genotype calls. Copy number alterations inferred from the hidden Markov model are indicated at the bottom of Figure 5. The segment means from the CBS are indicated by the black segments overlaying the copy number estimates. Approaches for calling deletions and amplifications from the segment means have been described elsewhere (Willenbrock and Fridlyand 2005).

```
R> stdev <- mad(df$cn, na.rm=TRUE)/100
R> redonfig <- xyplot(cn/100~position, df, pch=".", panel=cnPanel,
                      ylim=c(-0.5,6), ylab="total copy number",
                      pch.cols=genotype.cols,
                      gt=df$gt,
                      hmm.segs=fit.cn,
                      cbs.segs=cbs.segs1,
                      scales=list(x=list(tick.number=12)),
                      lwd=1,
                      shades=shades, key=mykey, xlim=c(0,150), draw.key=TRUE,
```

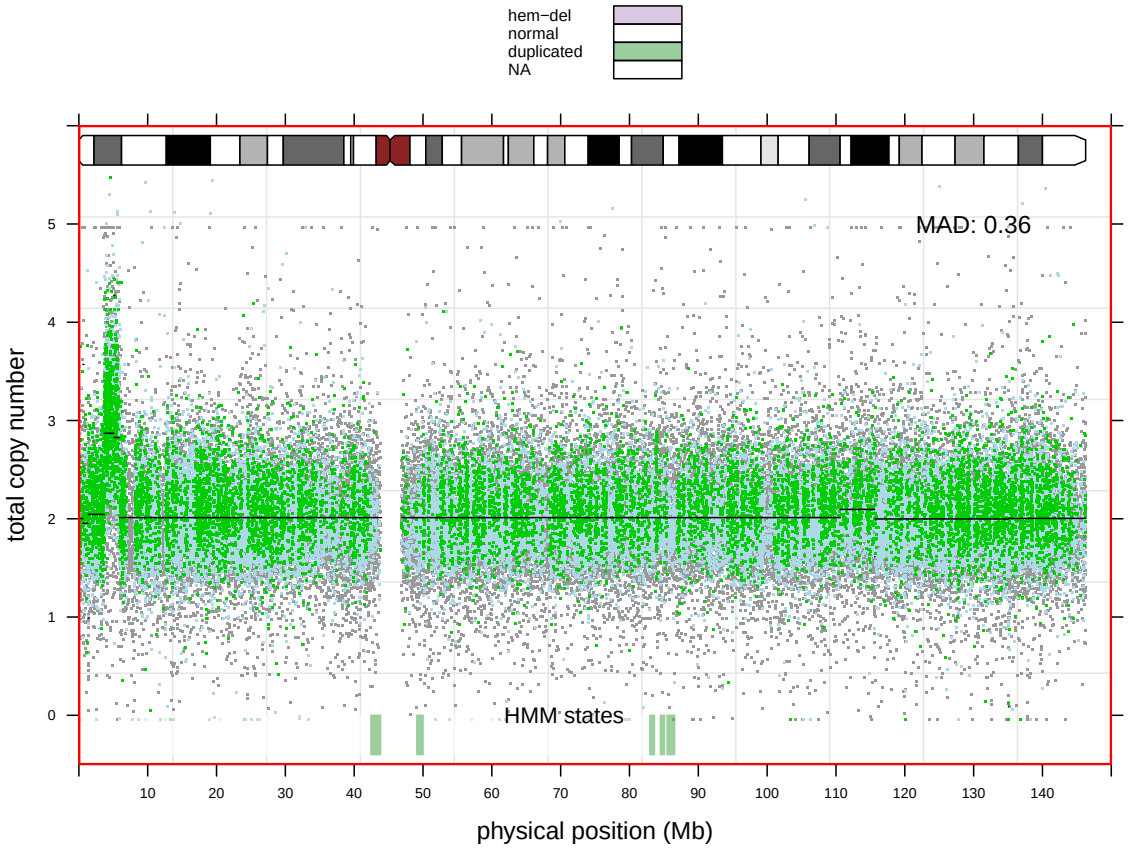

Figure 5: An amplification on the p-arm of chromosome 8 for HapMap sample NA19007. Estimates of raw copy number and the results of the HMM fit to the raw copy number. The segment means from circular binary segmentation overlay the copy number estimates. The colors of the plotting symbols indicate whether the CRLMM genotype is AA/BB (light blue) or AB (green). Nonpolymorphic markers are plotted in grey.

```
                    xlab="physical position (Mb)",
                    add.ideogram=TRUE,
                    par.strip.text=list(lines=0.9, cex=0.6))
R> print(redonfig)
R> trellis.focus("panel", 1, 1)
R> ltext(median(df$position), 0, "HMM states", cex=0.9)
```

Alternatives. Alternative packages for smoothing copy number estimates include the HMMs in the packages **Birdseye** (Korn *et al.* 2008) and **PennCNV** (Wang *et al.* 2007), and the breakpoint detection algorithms implemented in the packages **GLAD** (Hupe *et al.* 2004), **segclust** (Picard *et al.* 2005, 2007), and **GADA** (Pique-Regi *et al.* 2008).

5. Discussion

We have applied the **crlmm** software to the HapMap phase 3 data, illustrating the steps of preprocessing, the genotyping of polymorphic markers, and the estimation of allele-specific copy number. We organize the normalized intensities, statistical summaries from the genotyping and copy number estimation steps, and meta-data on the features and samples in a single container. This organization facilitates visualizations that allow inspection of the genotypes and copy number estimates in the context of the lower-level data. In addition, several of the algorithms have been adapted to allow parallelization and the underlying data structures currently implement utilities in the **ff** package to minimize **crlmm**'s memory footprint. We have provided several useful visualizations related to low-level copy number analysis using the HapMap data as an exemplar. Note that it would be straightforward to proceed in the opposite direction – to target genomic regions in which copy number estimates are associated with a particular phenotype, followed by more detailed inspection of the loci in the region.

Batch effects are common in large studies due to the extended period of time required to process the samples. The **crlmm** package models the variation driven by batch as part of the estimation procedure for copy number, permitting inference of copy number gain and loss from batch-adjusted locus-level summaries. We expect that such an approach will reduce the occurrence of spurious associations induced by temporal artifacts such as batch effects. Users should carefully inspect that the resulting inferences from the copy number analysis are not driven by technological artifacts. Future versions of **crlmm** may provide an interface with software specifically designed for batch detection, such as surrogate variable analysis implemented in the **sva** package (Leek and Storey 2007). Approaches for detecting and adjusting for batch effects have been described in a recent review (Leek *et al.* 2010).

While smoothing the locus-level estimates of copy number to infer regions of gain and loss is beyond the scope of the **crlmm** package, the ability to easily integrate with packages that provide these utilities is essential. Our analysis of the HapMap data illustrates a general workflow that begins with the raw fluorescence intensities from the array scanners and concludes with inferences of amplified regions and deletions from a hidden Markov model. The flexibility to tailor complex genomic analyses to specific use-cases, such as copy number inference in family-based studies, is a strength of the modular framework illustrated here.

6. Session information

The R package **crlmm** is available from Bioconductor `http://www.bioconductor.org`.

This document was prepared using Sweave. Computationally intensive steps, such as the genotype calling and copy number estimation were cached using the **cacheSweave** package (Peng and Eckel 2009).

- R version 2.15.0 Patched (2012-04-25 r59178), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.iso885915`, `LC_NUMERIC=C`, `LC_TIME=en_US.iso885915`, `LC_COLLATE=en_US.iso885915`, `LC_MONETARY=en_US.iso885915`, `LC_MESSAGES=en_US.iso885915`, `LC_PAPER=C`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.iso885915`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, methods, stats, tools, utils
- Other packages: Biobase 2.16.0, BiocGenerics 0.3.0, BiocInstaller 1.5.7, bit 1.1-8,

cacheSweave 0.6-1, crlmm 1.15.1, crlmmCompendium 1.1.0, DNAcopy 1.30.0, ellipse 0.3-7, ff 2.2-6, filehash 2.2-1, genefilter 1.38.0, IRanges 1.15.7, lattice 0.20-6, MASS 7.3-18, oligoClasses 1.19.8, RColorBrewer 1.0-5, SNPchip 2.2.0, stashR 0.3-5, VanillaICE 1.19.7

- Loaded via a namespace (and not attached): affyio 1.24.0, annotate 1.34.0, AnnotationDbi 1.18.0, Biostrings 2.24.1, codetools 0.2-8, compiler 2.15.0, DBI 0.2-5, digest 0.5.2, foreach 1.4.0, grid 2.15.0, iterators 1.0.6, msm 1.1, mvtnorm 0.9-9992, preprocessCore 1.18.0, RSQLite 0.11.1, splines 2.15.0, stats4 2.15.0, survival 2.36-14, xtable 1.7-0, zlibbioc 1.2.0

Acknowledgements

RBS was supported by grant R00HG005015 from the NIH/NHGRI and Johns Hopkins Medical Institutions' CTSA grant. IR was supported by NIH grant R01GM083084. RI was supported by NIH grant R01RR021967. We would like to think Marvin Newhouse for computing support.

References

Affymetrix (2006). "BRLMM: An Improved Genotype Calling Method for the Genechip Human Mapping 500k Array Set." *Technical report*, Affymetrix, Inc. White Paper.

Altug-Teber O, Dufke A, Poths S, Mau-Holzmann UA, Bastepe M, Colleaux L, Cormier-Daire V, Eggermann T, Gillessen-Kaesbach G, Bonin M, Riess O (2005). "A Rapid Microarray Based Whole Genome Analysis for Detection of Uniparental Disomy." *Hum Mutat*, **26**(2), 153–9. ISSN 1098-1004 (Electronic).

Bengtsson H, Irizarry R, Carvalho B, Speed TP (2008). "Estimation and Assessment of Raw Copy Numbers at the Single Locus Level." *Bioinformatics*, **24**(6), 759–767. doi:10.1093/bioinformatics/btn016. URL http://dx.doi.org/10.1093/bioinformatics/btn016.

Bolstad BM, Irizarry RA, Astrand M, Speed TP (2003). "A Comparison of Normalization Methods for High Density Oligonucleotide Array Data Based On Variance and Bias." *Bioinformatics (Oxford, England)*, **19**(2), 185–193. PUBM: Print; JID: 9808944; 0 (Molecular Probes); ppublish.

Carvalho B, Bengtsson H, Speed TP, Irizarry RA (2007). "Exploration, Normalization, and Genotype Calls of High-Density Oligonucleotide SNP Array Data." *Biostatistics*, **8**(2), 485–499. doi:10.1093/biostatistics/kxl042. URL http://dx.doi.org/10.1093/biostatistics/kxl042.

Carvalho BS, Louis TA, Irizarry RA (2010). "Quantifying Uncertainty in Genotype Calls." *Bioinformatics*, **26**(2), 242–249. doi:10.1093/bioinformatics/btp624. URL http://dx.doi.org/10.1093/bioinformatics/btp624.

Colella S, Yau C, Taylor JM, Mirza G, Butler H, Clouston P, Bassett AS, Seller A, Holmes CC, Ragoussis J (2007). "QuantiSNP: An Objective Bayes Hidden-Markov Model to Detect and Accurately Map Copy Number Variation Using SNP Genotyping Data." *Nucleic Acids Res*, **35**(6), 2013–2025. doi:10.1093/nar/gkm076. URL http://dx.doi.org/10.1093/nar/gkm076.

Consortium IH, Altshuler DM, Gibbs RA, Peltonen L, Altshuler DM, Gibbs RA, Peltonen L, Dermitzakis E, Schaffner SF, Yu F, Peltonen L, Dermitzakis E, Bonnen PE, Altshuler DM, Gibbs RA, de Bakker PIW, Deloukas P, Gabriel SB, Gwilliam R, Hunt S, Inouye M, Jia X, Palotie A, Parkin M, Whittaker P, Yu F, Chang K, Hawes A, Lewis LR, Ren Y, Wheeler D, Gibbs RA, Muzny DM, Barnes C, Darvishi K, Hurles M, Korn JM, Kristiansson K, Lee C, McCarrol SA, Nemesh J, Dermitzakis E, Keinan A, Montgomery SB, Pollack S, Price AL, Soranzo N, Bonnen PE, Gibbs RA, Gonzaga-Jauregui C, Keinan A, Price AL, Yu F, Anttila V, Brodeur W, Daly MJ, Leslie S, McVean G, Moutsianas L, Nguyen H, Schaffner SF, Zhang Q, Ghori MJR, McGinnis R, McLaren W, Pollack S, Price AL, Schaffner SF, Takeuchi F, Grossman SR, Shlyakhter I, Hostetter EB, Sabeti PC, Adebamowo CA, Foster MW, Gordon DR, Licinio J, Manca MC, Marshall PA, Matsuda I, Ngare D, Wang VO, Reddy D, Rotimi CN, Royal CD, Sharp RR, Zeng C, Brooks LD, McEwen JE (2010). "Integrating Common and Rare Genetic Variation in Diverse Human Populations." *Nature*, **467**(7311), 52–58. doi:10.1038/nature09298. URL http://dx.doi.org/10.1038/nature09298.

Fridlyand J, Snijders A, Pinkel D, Albertson D, Jain A (2004). "Hidden Markov Models Approach to the Analysis of Array CGH Data." *Journal of Multivariate Analysis*, **90**, 132–153.

Gentleman RC, Carey VJ, Bates DM, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J, Hornik K, Hothorn T, Huber W, Iacus S, Irizarry R, Leisch F, Li C, Maechler M, Rossini AJ, Sawitzki G, Smith C, Smyth G, Tierney L, Yang JY, Zhang J (2004). "Bioconductor: Open Software Development for Computational Biology and Bioinformatics." *Genome Biol*, **5**(10), R80.

Hua J, Craig DW, Brun M, Webster J, Zismann V, Tembe W, Joshipura K, Huentelman MJ, Dougherty ER, Stephan DA (2007). "SNIPer-HD: Improved Genotype Calling Accuracy by an Expectation-Maximization Algorithm for High-Density SNP Arrays." *Bioinformatics*, **23**(1), 57–63. doi:10.1093/bioinformatics/btl536. URL http://dx.doi.org/10.1093/bioinformatics/btl536.

Hupe P, Stransky N, Thiery JP, Radvanyi F, Barillot E (2004). "Analysis of Array CGH Data: From Signal Ratio to Gain and Loss of DNA Regions." *Bioinformatics*, **20**(18), 3413–3422. doi:10.1093/bioinformatics/bth418. URL http://dx.doi.org/10.1093/bioinformatics/bth418.

Irizarry RA, Hobbs B, Collin F, Beazer-Barclay YD, Antonellis KJ, Scherf U, Speed TP (2003). "Exploration, Normalization, and Summaries of High Density Oligonucleotide Array Probe Level Data." *Biostatistics*, **4**(2), 249–264. doi:10.1093/biostatistics/4.2.249. URL http://dx.doi.org/10.1093/biostatistics/4.2.249.

Karayiorgou M, Simon TJ, Gogos JA (2010). "22q11.2 Microdeletions: Linking DNA Structural Variation to Brain Dysfunction and Schizophrenia." *Nat Rev Neurosci*, **11**(6), 402–416. doi:10.1038/nrn2841. URL http://dx.doi.org/10.1038/nrn2841.

Korn JM, Kuruvilla FG, McCarroll SA, Wysoker A, Nemesh J, Cawley S, Hubbell E, Veitch J, Collins PJ, Darvishi K, Lee C, Nizzari MM, Gabriel SB, Purcell S, Daly MJ, Altshuler D (2008). "Integrated Genotype Calling and Association Analysis of SNPs, Common Copy Number Polymorphisms and Rare CNVs." *Nat Genet*, **40**(10), 1253–1260. doi:10.1038/ng.237. URL http://dx.doi.org/10.1038/ng.237.

Leek JT, Scharpf RB, Bravo HC, Simcha D, Langmead B, Johnson WE, Geman D, Baggerly K, Irizarry RA (2010). "Tackling the Widespread and Critical Impact of Batch Effects in High-Throughput Data." *Nat Rev Genet*, **11**(10), 733–739. doi:10.1038/nrg2825. URL http://dx.doi.org/10.1038/nrg2825.

Leek JT, Storey JD (2007). "Capturing Heterogeneity in Gene Expression Studies by Surrogate Variable Analysis." *PLoS Genet*, **3**(9), e161. doi:10.1371/journal.pgen.0030161. URL http://dx.doi.org/10.1371%2Fjournal.pgen.0030161.

Leisch F (2003). "Sweave and Beyond: Computations On Text Documents." In *Proceedings of the 3rd annual workshop on distributed statistical computing*.

McCarroll SA, Kuruvilla FG, Korn JM, Cawley S, Nemesh J, Wysoker A, Shapero MH, de Bakker PIW, Maller JB, Kirby A, Elliott AL, Parkin M, Hubbell E, Webster T, Mei R, Veitch J, Collins PJ, Handsaker R, Lincoln S, Nizzari M, Blume J, Jones KW, Rava R, Daly MJ, Gabriel SB, Altshuler D (2008). "Integrated Detection and Population-Genetic Analysis of SNPs and Copy Number Variation." *Nat Genet*, **40**(10), 1166–1174. doi:10.1038/ng.238. URL http://dx.doi.org/10.1038/ng.238.

Olshen AB, Venkatraman ES, Lucito R, Wigler M (2004). "Circular Binary Segmentation for the Analysis of Array-Based DNA Copy Number Data." *Biostatistics*, **5**(4), 557–72. doi:10.1093/biostatistics/kxh008. URL http://dx.doi.org/10.1093/biostatistics/kxh008.

Peng RD, Eckel SP (2009). "Distributed Reproducible Research Using Cached Computations." *Computing in Science & Engineering*, **11**(1), 28–34. doi:10.1109/MCSE.2009.6.

Picard F, Robin S, Lavielle M, Vaisse C, Daudin JJ (2005). "A Statistical Approach for Array CGH Data Analysis." *BMC Bioinformatics*, **6**(1), 27. doi:10.1186/1471-2105-6-27. URL http://dx.doi.org/10.1186/1471-2105-6-27.

Picard F, Robin S, Lebarbier E, Daudin JJ (2007). "A Segmentation/Clustering Model for the Analysis of Array CGH Data." *Biometrics*, **63**(3), 758–766. doi:10.1111/j.1541-0420.2006.00729.x. URL http://dx.doi.org/10.1111/j.1541-0420.2006.00729.x.

Pinto D, Pagnamenta AT, Klei L, Anney R, Merico D, Regan R, Conroy J, Magalhaes TR, Correia C, Abrahams BS, Almeida J, Bacchelli E, Bader GD, Bailey AJ, Baird G, Battaglia A, Berney T, Bolshakova N, Bölte S, Bolton PF, Bourgeron T, Brennan S, Brian J, Bryson SE, Carson AR, Casallo G, Casey J, Chung BHY, Cochrane L, Corsello C, Crawford EL, Crossett A, Cytrynbaum C, Dawson G, de Jonge M, Delorme R, Drmic I, Duketis E, Duque F, Estes A, Farrar P, Fernandez BA, Folstein SE, Fombonne E, Freitag CM, Gilbert J, Gillberg C, Glessner JT, Goldberg J, Green A, Green J, Guter SJ, Hakonarson H, Heron EA, Hill M, Holt R, Howe JL, Hughes G, Hus V, Igliozzi R, Kim C, Klauck SM, Kolevzon

A, Korvatska O, Kustanovich V, Lajonchere CM, Lamb JA, Laskawiec M, Leboyer M, Couteur AL, Leventhal BL, Lionel AC, Liu XQ, Lord C, Lotspeich L, Lund SC, Maestrini E, Mahoney W, Mantoulan C, Marshall CR, McConachie H, McDougle CJ, McGrath J, McMahon WM, Merikangas A, Migita O, Minshew NJ, Mirza GK, Munson J, Nelson SF, Noakes C, Noor A, Nygren G, Oliveira G, Papanikolaou K, Parr JR, Parrini B, Paton T, Pickles A, Pilorge M, Piven J, Ponting CP, Posey DJ, Poustka A, Poustka F, Prasad A, Ragoussis J, Renshaw K, Rickaby J, Roberts W, Roeder K, Roge B, Rutter ML, Bierut LJ, Rice JP, Salt J, Sansom K, Sato D, Segurado R, Sequeira AF, Senman L, Shah N, Sheffield VC, Soorya L, Sousa I, Stein O, Sykes N, Stoppioni V, Strawbridge C, Tancredi R, Tansey K, Thiruvahindrapduram B, Thompson AP, Thomson S, Tryfon A, Tsiantis J, Engeland HV, Vincent JB, Volkmar F, Wallace S, Wang K, Wang Z, Wassink TH, Webber C, Weksberg R, Wing K, Wittemeyer K, Wood S, Wu J, Yaspan BL, Zurawiecki D, Zwaigenbaum L, Buxbaum JD, Cantor RM, Cook EH, Coon H, Cuccaro ML, Devlin B, Ennis S, Gallagher L, Geschwind DH, Gill M, Haines JL, Hallmayer J, Miller J, Monaco AP, Jr JIN, Paterson AD, Pericak-Vance MA, Schellenberg GD, Szatmari P, Vicente AM, Vieland VJ, Wijsman EM, Scherer SW, Sutcliffe JS, Betancur C (2010). "Functional Impact of Global Rare Copy Number Variation in Autism Spectrum Disorders." *Nature*. doi:10.1038/nature09146. URL http://dx.doi.org/10.1038/nature09146.

Pique-Regi R, Monso-Varona J, Ortega A, Seeger RC, Triche TJ, Asgharzadeh S (2008). "Sparse Representation and Bayesian Detection of Genome Copy Number Alterations From Microarray Data." *Bioinformatics*, **24**(3), 309–318. doi:10.1093/bioinformatics/btm601. URL http://dx.doi.org/10.1093/bioinformatics/btm601.

R Development Core Team (2011). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Rabbee N, Speed TP (2006). "A Genotype Calling Algorithm for Affymetrix SNP Arrays." *Bioinformatics*, **22**(1), 7–12. doi:10.1093/bioinformatics/bti741. URL http://dx.doi.org/10.1093/bioinformatics/bti741.

Redon R, Ishikawa S, Fitch KR, Feuk L, Perry GH, Andrews TD, Fiegler H, Shapero MH, Carson AR, Chen W, Cho EK, Dallaire S, Freeman JL, Gonzalez JR, Gratacos M, Huang J, Kalaitzopoulos D, Komura D, MacDonald JR, Marshall CR, Mei R, Montgomery L, Nishimura K, Okamura K, Shen F, Somerville MJ, Tchinda J, Valsesia A, Woodwark C, Yang F, Zhang J, Zerjal T, Zhang J, Armengol L, Conrad DF, Estivill X, Tyler-Smith C, Carter NP, Aburatani H, Lee C, Jones KW, Scherer SW, Hurles ME (2006). "Global Variation in Copy Number in the Human Genome." *Nature*, **444**(7118), 444–454. ISSN 1476-4687 (Electronic). doi:10.1038/nature05329.

Ritchie ME, Carvalho BS, Hetrick KN, Tavaré S, Irizarry RA (2009). "R/Bioconductor Software for Illumina's Infinium Whole-Genome Genotyping BeadChips." *Bioinformatics*, **25**(19), 2621–2623. doi:10.1093/bioinformatics/btp470. URL http://dx.doi.org/10.1093/bioinformatics/btp470.

Sarkar D (2008). *Lattice: Multivariate Data Visualization with R*. Springer-Verlag, New York. ISBN 978-0-387-75968-5.

Scharpf RB, Parmigiani G, Pevsner J, Ruczinski I (2008). "Hidden Markov Models for the Assessment of Chromosomal Alterations Using High-Throughput SNP Arrays." *Annals of Applied Statistics*, **2**(2), 687–713. URL http://dx.doi.org/10.1214/07-AOAS155.

Scharpf RB, Ruczinski I, Carvalho B, Doan B, Chakravarti A, Irizarry RA (2011). "A Multilevel Model to Address Batch Effects in Copy Number Estimation Using SNP Arrays." *Biostatistics*, **12**(1), 33–50. doi:10.1093/biostatistics/kxq043. URL http://dx.doi.org/10.1093/biostatistics/kxq043.

Venkatraman ES, Olshen AB (2007). "A Faster Circular Binary Segmentation Algorithm for the Analysis of Array CGH Data." *Bioinformatics*, **23**(6), 657–663. doi:10.1093/bioinformatics/btl646. URL http://dx.doi.org/10.1093/bioinformatics/btl646.

Viterbi A (1967). "Error Bounds for Convolution Codes and an Asymptotically Optimal Decoding Algorithm." *IEEE Transactions on Information Theory*, **13(2)**, 260–269.

Wang K, Li M, Hadley D, Liu R, Glessner J, Grant SFA, Hakonarson H, Bucan M (2007). "PennCNV: An Integrated Hidden Markov Model Designed for High-Resolution Copy Number Variation Detection in Whole-Genome SNP Genotyping Data." *Genome Res*, **17**(11), 1665–1674. doi:10.1101/gr.6861907. URL http://dx.doi.org/10.1101/gr.6861907.

Wang W, Carvalho B, Miller N, Pevsner J, Chakravarti A, Irizarry RA (2008). "Estimating Genome-Wide Copy Number Using Allele Specific Mixture Models." *Journal of Computational Biology*, **15**(7), 857–866.

Wellcome Trust Case Control Consortium (2007). "Genome-Wide Association Study of 14,000 Cases of Seven Common Diseases and 3,000 Shared Controls." *Nature*, **447**(7145), 661–678. ISSN 1476-4687 (Electronic). doi:10.1038/nature05911.

Willenbrock H, Fridlyand J (2005). "A Comparison Study: Applying Segmentation to Array CGH Data for Downstream Analyses." *Bioinformatics*, **21**(22), 4084–4091. doi:10.1093/bioinformatics/bti677. URL http://dx.doi.org/10.1093/bioinformatics/bti677.

Wu Z, Irizarry R (2005). "Stochastic Models Inspired by Hybridization Theory for Short Oligonucleotide Arrays." *Journal of Computational Biology*, **12**(6), 882–893. URL http://www.liebertonline.com/doi/abs/10.1089/cmb.2005.12.882.

Zhang Q, Ding L, Larson DE, Koboldt DC, McLellan MD, Chen K, Shi X, Kraja A, Mardis ER, Wilson RK, Boreki IB, Province MA (2009). "CMDS: A Population-Based Method for Identifying Recurrent DNA Copy Number Aberrations in Cancer From High-Resolution Data." *Bioinformatics*. doi:10.1093/bioinformatics/btp708. URL http://dx.doi.org/10.1093/bioinformatics/btp708.

Work in progress

B allele frequencies and parallelization

One can compute B allele frequencies and log R ratios for each of the samples and use these as

inputs to the HMM. The B allele frequencies can be particularly helpful for amplifications. In the following unevaluated code chunk we create a *oligoSnpSetList* object where each element in the list is an *oligoSnpSet* for one chromosome. As only two chromosomes are specified in the following code chunk, the list has length two.

```
R> oligoSetList <- cnSet2oligoSetList(cnSet, batch.name="SHELF", chrom=c(8,9))
R> save(oligoSetList, file="../../data/oligoSetList.rda")

R> print(oligoSetList)

oligoSetList of length 2

R> chromosome(oligoSetList)

[1] "8" "9"

R> ls(assayData(oligoSetList))

[1] "baf"             "call"            "callProbability"
[4] "copyNumber"
```

A HMM incorporating the B allele frequencies and log R ratios can be fit using the `hmm` method. When used with a package supporting parallel processing, such as the R package **snow**, the HMM will automatically be parallelized over elements (chromosomes) in the `oligoSnpSetList` object.

```
R> fit2 <- hmm(oligoSetList, sampleIds="NA19007")
```

Visualization of RangedData and low-level summaries with lattice

```
R> fitAltered <- fit2[state(fit2) %in% c(1,2,5,6) & chromosome(fit2)==8,]
R> oligoSet <- oligoSetList[[1]]
R> fig <- xyplotLrrBaf(rd=fitAltered, object=oligoSet,
                       frame=1e6, panel=SNPchip:::xypanelBaf,
                       ylim=c(-4, 1.5),
                       cex=0.2,
                       pch=21,
                       col.het="red",
                       fill.het="red",
                       col.hom="grey",
                       fill.hom="grey",
                       state.cex=0.8,
                       border="orange", scales=list(x="free"),
                       par.strip.text=list(cex=0.5),
                       xlab="Mb", ylab=expression(log[2]("R ratio")))
```

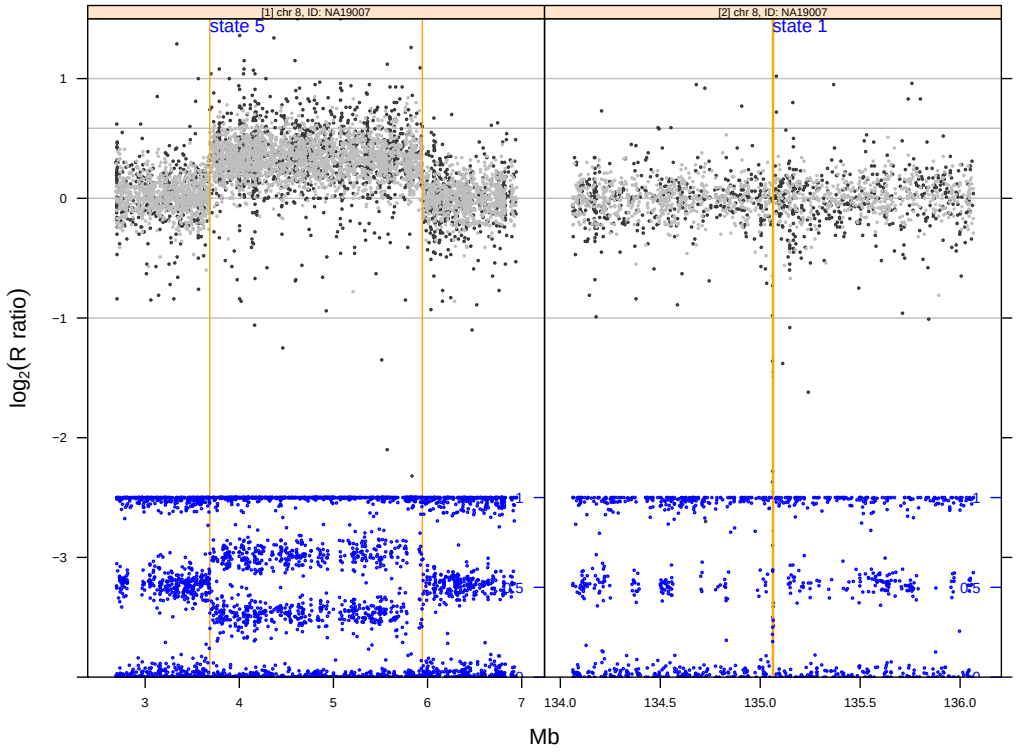

Figure 6: Plot of log R ratios and BAFs for alterations on chromosome 8.

```
pdf
  2
```

Affiliation:

Robert Scharpf
Department of Oncology,
Johns Hopkins University School of Medicine,
550 N. Broadway, Suite 1103
Baltimore, MD 21218
E-mail: rscharp1@jhmi.edu

Rafael Irizarry and Ingo Ruczinski
Department of Biostatistics
Johns Hopkins Bloomberg School of Public Health
615 North Wolfe Street
Baltimore MD 21218
E-mail:rafa@jhu.edu and ingo@jhu.edu

Matthew Ritchie
Bioinformatics Division
The Walter and Eliza Hall Institute of Medical Research
1G Royal Parade
Parkville, Victoria 3052
Australia E-mail: mritchie@wehi.edu.au

Benilton Carvalho
Department of Oncology
University of Cambridge
CRUK Cambridge Research Institute
Li Ka Shing Centre
Robinson Way
Cambridge CB2 ORE
United Kingdom E-mail: Benilton.Carvalho@cancer.org.uk

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume VV, Issue II *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd