CHAPTER 2

DEVELOPMENT OF A CNV DISCOVERY PIPELINE FOR AFFYMETRIX 6.0

2.1 Introduction

2.1.1 CNV discovery using microarrays

There are two major types of data that serve as the source of CNV discovery using microarrays: two-channel array-Comparative Genomic Hybridization (CGH) and genotyping arrays. The difference in the nature of the array affects data normalization, the models underpinning CNV discovery algorithms and interpretation of results.

Array-CGH hybridizes two differentially labeled DNA samples, often one test sample and one reference sample, together on the same array and the difference in DNA dosage between the two samples is reflected by the difference in fluorescent intensity between the two channel. A log ratio of the intensity is often calculated for each probe and its significant deviation from zero is an indication of copy number differences between the test and reference samples in regions targeted by the corresponding probes in the test sample relative to the reference. For this type of data, technical variation among different probes is internally controlled. Algorithms essentially find outliers in ratio space. However, the derived copy number difference

is always relative and the choice of the reference affects the translation of relative copy number difference into absolute copy number.

Genotyping arrays are primarily composed of pairs of oligonucleotide probes that target the same locus but different alleles of each selected SNPs. Only one DNA sample is hybridized to the array and DNA dosage is reflected by the intensity of the probes and also partially/indirectly by the intensity ratio between the two probes targeting the same SNP. Some genotyping arrays also contain non-variable probes similar to array-CGH probes, which only provide intensity information that reflects absolute DNA dosage due to the single-channel design. For this type of data, technical variation among different probes needs to be removed explicitly in data analysis. Algorithms work in intensity space and absolute copy number can in principle be determined once probe dosage response has been calibrated.

2.1.1.1 Affymetrix Genome-wide human SNP array 6.0

Affymetrix genome-wide human SNP array 6.0 (Affy6) is an array platform that aims to perform both high-density SNP genotyping and high resolution CNV discovery simultaneously. It was developed between Affymetrix and the Broad Institute [18]. The array has 906,600 SNP probe sets and 946,000 copy number probe sets. The latter includes 202,000 probe sets targeting 5,677 CNV regions from the Database of Genomic Variants at high density and the rest spread evenly along the genome [19]. Each SNP probe set contains multiple oligonucleotide features that are identical copies of one of the two probes targeting the two possible alleles. Each copy number probe set contains multiple identical features targeting the same genomic location 1. For simplicity, I will use 'probe' to refer to 'probe set' when describing analyses that use only summarized probe set intensities or their derivatives.

After hybridization, washing and scanning, a .CEL file is produced for each sample genotyped with Affy6, which contains information including probe locations and intensities. Affymetrix developed a suite of command line tools called Affymetrix Power Tools (APT) [20] for extracting information from the .CEL files and common downstream analysis such as SNP calling and CNV discovery. A number of CNV calling methods can also be applied to Affy6 data once the probe intensities have

been extracted by APT.

2.1.2 CNV discovery algorithms

Regardless of the type of the array, the typical data summary that is input into CNV discovery algorithms is often a sequence of values (intensity or log ratio) with ordered spatial coordinates along a chromosome. For genotyping arrays, a second sequence of values (measuring the relative intensity of the two alleles, often called 'B allele frequency') sharing the same spatial coordinates with the first sequence of values is available. CNV discovery aims to solve the problem of finding spatial segments with values sufficiently different from adjacent segments as a result of belonging to one of a finite set of copy number states that is different between adjacent segments.

Many CNV discovery methods have been developed. Except for a few methods that use empirical cut-off values [21, 22] or hierarchical clustering [23], most of them can be placed into one of the following two broad categories: segmentation-based (change-point-finding) methods and hidden-markov-model-based (HMM-based) methods.

2.1.2.1 Segmentation-based methods

This category of methods search for change points in an ordered sequence of values that define segments having different distribution of values (often measured by having different means). Circular binary segmentation is a typical method belonging to this category proposed by Olshen *et al* [24] that recursively test if a new segment or breakpoint should be introduced inside an existing segment based on the differences in the distribution of values between the newly introduced segment and the rest of the existing segment or between the two resulting segments separated by the proposed breakpoint. Jong *et al* [25] proposed a method that models the values along a chromosome as a sequence of normal distributions with different parameters and used the genetic algorithm to find the spatial boundaries that maximize the likelihood that actual values are drawn from those normal distributions. Similarly,

Hupe *et al* [26] modeled segments of different copy number states along a chromosome as a piecewise constant function and estimated the parameters of the function using adaptive weights smoothing. Pique-Regi *et al* [27] further formulated piecewise constant function as linear combinations of step functions and used sparse Bayesian learning (SBL) to obtain the best linear combination that fits the data. All segmentation-based methods have certain measures to restrict over-segmentation during maximization of model fitting. This usually involves substituting log likelihood with Akaike Information Criterion (AIC) or Bayes Information Criterion (BIC) or similar criteria as the target function for optimization to penalize the use of more parameters [25–27] and/or a separate pruning step after segmentation is done to eliminate spurious breakpoints [27].

2.1.2.1.1 GADA GADA is the implementation for the method described in [27]. It has a SBL step that provides initial breakpoints of which level of sparseness is controlled by the parameter a (the larger the more sparse) and a backward elimination step that removes spurious breakpoints of which stringency is controlled by the parameter T (the larger the more stringent).

2.1.2.2 HMM-based methods

HMM-based methods model the ordered sequence of values as a sequence of observed states that are determined by a chain of discrete hidden states, each one of which is determined probabilistically by its previous hidden state(s). The key parameters of a HMM include the number of hidden states K, the vector of initial state probability π, the state transition probability matrix A and collection of emission probability functions B. Fridlyand *et al* [28] applied unsupervised HMM to array-CGH data. B was assumed to be a collection of Gaussian distributions, each corresponding to a hidden state, and the initial parameters of B were estimated through clustering. Parameters (π, A, B) were optimized using the EM algorithm. The number of states K was chosen to minimize an AIC-like criteria to balance between model fitting and restricting the total number parameters. Finally, The states were merged into segments with user-defined criteria. Marioni *et al* [29] improved

the model by using distance-aware transition probabilities to account for heterogeneity in probe density. Guha *et al* [30] modified the model to use a fixed 4-states HMM and incorporated Bayesian learning in which each state represented a predefined copy number state, informative priors were imposed on model parameters and MCMC was used in learning model parameters and generating copy number states. In this way, segmentation and classification was performed simultaneously. Shah *et al* [31] modeled the emission probability distribution of each state as a mixture of two Gaussian distributions with one component representing values generated from the given state and the other representing outliers, which improved the robustness of CNV calling. Methods designed for SNP genotyping arrays further exploit the additional B allele frequency (BAF) information. QuantiSNP is an objective Bayes HMM-based algorithm highly tailored to Illumina Beadarray data [32]. Similar to Marioni *et al* and Shah *et al*, state transition probabilities were adjusted for local probe distance and outliers were considered in modeling emission probabilities. Emission probabilities for BAF were modeled alongside log R ratio. Parameters were estimated using the EM algorithm with hyper-parameters of the conjugate priors for the emission model estimated from a reference dataset with known copy number. The program calculates a Bayes factor for each CNV called that facilitates ranking and post-filtering of CNV calls. PennCNV is another widely used HMM-based program for the CNV analysis of Illumina Beadarray data [33]. Its underlying model is very similar to QuantiSNP, except it incorporates population B allele frequency in the emission model for BAF and it has an *a posteriori* validation step using family information if available.

2.1.2.2.1 Birdsuite Birdsuite is a software suite highly tailored to the Affy6 data that integrates SNP and CNV calling. Its CNV discovery component, Birdseye, is an HMM-based program. Unlike most HMM-based methods, Birdseye receives predefined parameters for emission probability distributions from Canary and Birdseed, the components of Birdsuite that run prior to Birdseye that estimate copy number at known CNVs and genotype SNPs respectively. Those parameters are probe-specific and are estimated using the EM algorithm during the running of Canary and Birdseed with priors learned from samples of known genotype. The state

transition probabilities are also pre-defined, distance-dependent and tuned to the probe density of Affy6. Birdseye uses the Viterbi algorithm to determine the most probable chain of states and produces a LOD score for each segment representing the strength of evidence [18].

2.1.3 CNV calling pipeline

CNV calling algorithms solve the specific problem of identifying genomic segments with likely aberrant copy number from input sequences of values with ordered spatial coordinates. However, the process that takes raw data generated by microarray experiments and produces CNV calls ready for downstream analysis involves many other steps that can affect the quality of the final set of CNV calls remarkably. The structure of a typical CNV calling pipeline is demonstrated in Figure 2.1. After raw intensities have been extracted, a pre-processing step is usually mandatory prior to CNV calling. In this step, various normalization procedures may be applied to remove technical biases or variation between channels of an array, across probes of different spatial location or genomic context, or across different array experiments, etc. Normalized intensities may be organized and transformed to the format required by the calling algorithms. Technical failures may also be identified and removed at this stage. After intensities have been properly normalized and transformed, multiple calling algorithms may be applied to complement or support one another. Next, resultant CNV calls are subjected to post-processing that usually involves computing quality control (QC) metrics and filtering calls and samples based on those QC metrics. Additional procedures such as merging CNV calls may be necessary depending on the calling algorithm that has been applied. Various visualization tools are often an essential part of the pipeline that facilitates quality control and the selection of filtering thresholds. It is crucial to assess the performance of such pipeline with an independent CNV dataset, ideally of higher quality and from the same samples, based on which the pipeline may be optimized.

The extent of completeness in implementing the above pipeline varies among current CNV calling programs. Some have pre-processing capabilities, such as the APT utility apt-copynumber-workflow, which handles intensity extraction, normaliza-

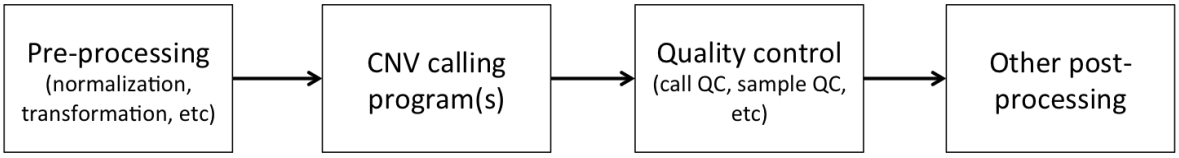

Figure 2.1: Simplified diagram of a typical CNV calling pipeline

tion, probe set summarization and CNV calling. Some are dedicated CNV callers that do not implement any pre- or post-processing at all, such as GADA, which simply outputs segmentation on receiving an input sequence of log ratios. The usability of output CNV calls also varies. Again taking apt-copynumber-workflow as an example, instead of delivering genomic segments with copy number, it only outputs the inferred copy number state of every probe set. GADA provides genomic segments but does not assign copy number state. Birdseye provides the most usable calls of the three, as it not only produces genomic segments with copy number state, but also produces the statistical confidence of the called CNVs that facilitates post-processing. Regardless of the above differences, current CNV programs provide little post-processing and QC functions, whereas robust and consistent post-processing is of vital importance in producing a reliable CNV call set, especially in studies with a large sample size and/or multiple datasets. A few simple post-processing methods have been applied in previous CNV studies [33, 34], which were limited to filtering CNV calls by number of probes and size or removing samples with large variance in probe intensities or apparent mosaic chromosomes.

In the result section of this chapter, I will first compare the performance of three CNV calling programs on Affy6 data. I will next describe a CNV pipeline I developed for Affy6 data that features an effective sample QC. Finally, I will demonstrate the application of this pipeline to several Affy6 datasets that will be further discussed in later chapters. The implementation details are provided in the methods section.

2.2 Materials and methods

2.2.1 Extracting probe intensities and re-producing the scanned image

Raw probe intensities and probe IDs were extracted from .CEL files using the APT command 'apt-cel-extract'. The positions of a probe on the array can be derived from its probe ID using the equations provided by Affymetrix [20]:

$$x = (probeID - 1) \bmod N_{\text{column}}$$

$$y = \text{floor}\left((probeID - 1) / N_{\text{column}}\right)$$

where, N_{column} stands for the number of columns of the probe array, which is 2680 for Affy6 [20]. I wrote an R script to calculate the positions, re-order the probes according to their positions and plot the scanned image using heat map colors. The brighter the color, the higher the intensities.

2.2.2 Extracting and normalizing probe set intensities

I extracted probe set intensities from .CEL files and normalized them across samples on the same sample plate using the APT command 'apt-probeset-summarize' with the option 'quant-norm.target=1000,pm-only,plier.optmethod=1,expr.genotype=true'. This command first extracted probe intensities from all input sample .CEL files, then applied quantile normalization to adjust all samples to the same distribution with a median probe intensity value of 1000 and lastly summarize probe set intensities from composing probes using the PLIER (probe logarithmic intensity error) estimation with the 'perfect match only' option [20].

2.2.3 Transform probe set intensities into log ratios

Let $x_{i,j}$ denotes the summarized and normalized intensity of probe set i in sample j. The log ratio $y_{i,j}$ was calculated as:

$$y_{i,j} = \log_2 \frac{x_{i,j}}{\text{median}\,(x_{i,*})}$$

, where median $(x_{i,*})$ is the median value of all samples on the same plate.

2.2.4 Calculating log-ratio-related sample QC statistics

Noise level and extent of spatial waviness (autocorrelation) of array data are two important factors that remarkably affect CNV analysis as will be described later. I used median absolute deviation (MAD) of probe sets log ratios as the measure of noise level and sum of auto-correlation (SAC) along the chromosomes as the measure of spatial waviness. For sample j:

$$MAD_j = \text{median}\left(|y_{*,j} - \text{median}(y_{*,j})\,|\right)$$

$$SAC_j = \sum_{k=1}^{n=5} \left| \frac{\mathrm{E}\left[(Y_{i,j} - \mu_{y_j})(Y_{i+k,j} - \mu_{y_j})\right]}{\sigma^2_{y_j}} \right|$$

2.2.5 Correction for spatial auto-correlation

For each sample, correction was done by chromosomes using the method developed by [35]. Briefly, a loess curve was fitted to the $\log_2$ ratios along a chromosome with a window size containing 10% of the probes in the chromosome and the $\log_2$ ratios were replaced by the residuals (Figure 2.2).

2.2.6 Storage and retrieval of normalized intensity data

Normalized probe set intensities were stored as a probe-set-by-sample table with probe set name and chromosomal location in HDF format [36]. Each table contained

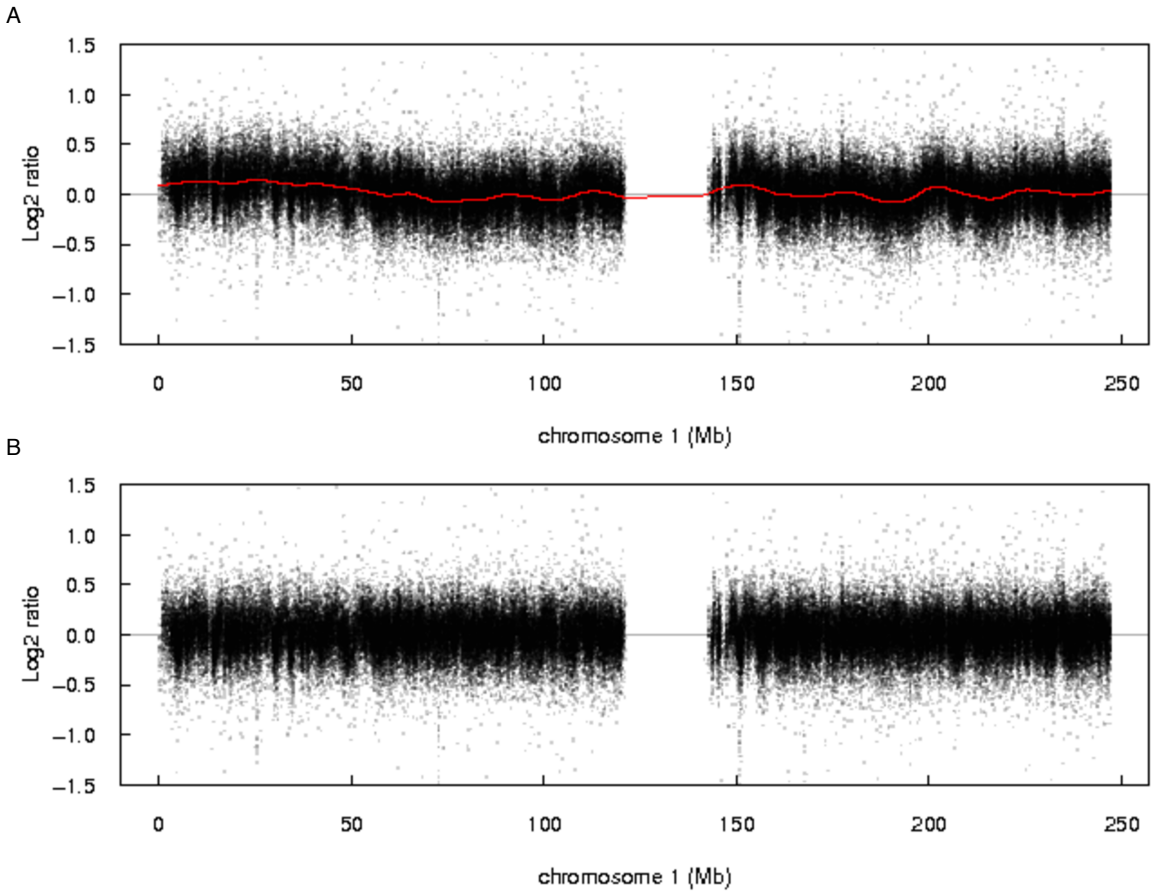

Figure 2.2: Demonstration of correcting spatial auto-correlation, showing $i\log_2$ ratio profile across chromosome 1 of a sample with a SAC of 0.7766 (top 0.15%) before (A) and after (B) correction for spatial auto-correlation. The red curve in (A) is the loess curved fitted with a window size of 14k probes (10% of all probes targeting chromosome 1)

a plate of samples. I developed a python utility using the PyTables package [37] to write such table in HDF format and create an index in the column containing chromosomal locations. The resulting HDF file was similar to an in-file database. I also wrote a python utility for retrieving probe set intensities by chromosomal coordinates from such files.

2.2.7 The CNV call format

This is the format of plain text files in which CNV calls were recorded. Each CNV is described in one row, of which fields are separated by tab and the first seven fields

are required. The required fields are chromosome, start coordinate, end coordinate, number of probes contained, average log ratio, sample ID and copy number change. Additional fields may be appended to the end of row.

2.2.8 Merging split CNV calls

CNV calls on the same chromosome were sorted by genomic coordinates and scanned through, each time taking a pair of adjacent calls. The two adjacent calls were merged into one, if:

1. Both calls have the same genotype
2. The number of probes separating the two calls < 100
3. The ratio of the number of probes separating the two calls to the number of probes in the merged call $< 10\%$
4. The probe density between the two calls > 1 probe per 5kb
5. The absolute difference in average $\log_2$ratio between the two calls < 0.15

The scan and merge process was repeated until no CNV calls could be merged.

2.2.9 CNVE clustering

To combine CNVs called in different individuals into CNV events (CNVEs), I used a hierarchical-clustering-like method described in [12]. Briefly, pairwise reciprocal overlap (RO) were first calculated among CNVs overlapping at least 1bp and CNV pairs with greatest RO were merged into a CNVE if RO>50%. Then, unmerged CNVs having a RO>50% with all CNVs already merged into this CNVE were iteratively merged in order of best RO. This method guarantees that the ROs between all pairs of CNVs belonging to a CNVE are greater than 50% and when a CNV has RO>50% with CNVs of multiple CNVEs, it is merged to the one with better RO. The boundaries of a CNVE were defined as those enclosing the minimum genomic interval that encompasses 90% of belonged CNVs.

2.2.10 Definition for different overlap criteria

Given two genomic intervals A and B on the same chromosome defined by coordinates $[start_A, end_A]$ and $[start_B, end_B]$, the length of overlap $L = \min(end_A, end_B) - \max(start_A, start_B) + 1$. Simple overlap is defined as $L > 0$. Overlap relative to interval A is: $O_A = L/(end_A - start_A + 1)$. Overlap interval $A > 50\%$ is defined as $O_A > 0.5$. Reciprocal overlap $> 50\%$ is defined as: $O_A > 0.5$ and $O_B > 0.5$.

2.2.11 Heuristic quality score for APT and GADA CNV calls

The quality score Q is defined as $N_{probe} \times \left|\overline{logRatio}\right|$.

2.3 Results

2.3.1 Comparing discovery programs for Affy6 data

2.3.1.1 Test pipeline for assessing CNV calling programs

To test the performance of apt-copynumber-workflow, GADA and Birdsuite, three test pipelines were constructed.

As stated in section 2.1.3, apt-copynumber-workflow is a fairly standalone program that handles both pre-processing (intensity extraction, normalization, probe set summarization, transformation into log ratio) and CNV discovery. However, it only produces copy number state for individual probe sets. Therefore, an extra step was added to merge adjacent probe sets into one CNV call if they had the same copy number state and their copy number were not equal to 2 and to calculate other information such as average log ratio that were required by the CNV call format (as described in section 2.2.7). Then CNVs were filtered by size, number of probes, and probe density and samples with excessive CNV calls were removed.

Since GADA is a dedicated CNV caller, apt-probeset-summarize was used to handle intensity extraction, normalization and probe set summarization. Probe set intensities were then transformed into log ratio as described in method. Unlike apt-copynumber-workflow, wherein intervention is not possible between pre-processing and CNV calling, an extra step that corrects spatial auto-correlation was added before running GADA, as it reduces the long range waviness in the data (Figure 2.2). Since GADA only performs segmentation but not copy number assignment, thresholds were applied to the distribution of average log ratio of segments to distinguish CNV calls and segments with normal copy number. The resulting CNV calls were stored in CNV call format and filtered using the same criteria as for calls made by apt-copynumber-workflow. Samples with excessive CNV calls were also removed.

Birdsuite calls apt-probeset-summarize to handle pre-processing. Since it works with probe set intensities instead of log ratios, transformation was not needed. Only output from the Birdseye algorithm were passed on for downstream analyses as Canary performs CNV typing at known CNVs rather than CNV discovery. The Birds-

eye calls were filtered using the LOD score in addition to the same criteria as above and samples with excessive CNV calls were removed.

2.3.1.2 Comparing general characteristics of call sets

I used the above test pipelines to call CNVs in 270 HapMap1 individuals. A number of program parameters and filter parameters were tested as listed in Table 2.1. I examined the median number of calls per sample, the median size of calls, the number of CNVEs, the fraction of singletons and overlap with published CNV datasets [12, 34] (Table 2.1).

Table 2.1: **Summaries of call sets produced by test piplines**

Program	Program or filter parameter	Median #call per sample	Median call size (kb)	Deletion-to-duplication ratio	#CNVE	%Singleton CNVE	%Overlap rate*	%Overlap rate†
APT	Default	64	14.6	2.82	2861	49.2	27.2	25.5
GADA	a=1 T=7 M=5‡	88	14.0	2.54	4514	49.3	23.8	23.2
GADA	a=1 T=8 M=5	73	15.2	2.61	3500	47.7	27.9	26.1
GADA	a=1 T=9 M=5	61	17.3	2.67	2833	47.2	30.0	27.9
GADA	a=1 T=10 M=5	51	19.4	2.80	2307	45.8	31.8	29.6
Birdseye	LOD≥5	86	14.8	4.03	3469	48.5	30.9	24.7
Birdseye	LOD≥10	59	23.4	4.20	2176	46.9	35.3	28.6

* Proportion of calls reciprocally overlapped by common CNVs described in McCarroll *et al* [34].

† Proportion of calls reciprocally overlapped by ng42M CNVs described in Conrad *et al* [12].

‡ For a and T see section 2.1.2.1.1; M defines the minimal required number of probes.

There were a few characteristics that were shared by or similar among all pipelines. For example, (*i*) all pipelines called 50–90 CNVs per individuals, (*ii*) the median call size of the majority of call sets were all in the range of 15–20kb, (*iii*) the proportions of singletons were close to 50%, (*iv*) one quarter to one third of the CNVEs were found previously, (*v*) all three CNV calling methods were better at calling large recurrent deletions as indicated by increased call size and deletion-to-duplication ratio and decreased proportion of singletons with increasing stringency (GADA T7 to T10, Birdseye LOD5 to LOD10) of calling. The deletion-to-duplication ratio differed remarkably between call sets produced by Birdseye and the other two programs. This is likely due to that Birdseye calls from intensities whereas the other two calls from log ratios (see section 2.4.2).

2.3.1.3 Benchmark by a high quality call set

Comparing with one or more independent high quality datasets generated from the same samples can provide direct assessment of the performance of the calling algorithms. Previously, a set of tiling resolution CNV calls were produced from 40 individuals, 19 of which were from the HapMap1 individuals, using a set of Nimblegen CGH arrays that collectively contained 42M probes [12] (referred to as ng42M). I used this dataset as a gold standard to benchmark GADA T9, GADA T10 and Birdseye call sets, as the rest of the call sets were apparently of lower quality. The fraction of ng42M CNVs reciprocally overlapped >50% by test call set in the same individual was used as a measure of sensitivity and the fraction of test call set overlapped by ng42M CNVs in the same individual was used as a measure of specificity (Table 2.2–2.9).

Breaking down by CNV size, sensitivity generally increased as call size increased in all three call sets as expected. Specificity, however, was highest in the middle ranges (10kb to 100kb) and sharply dropped to roughly 10% for CNVs above 500kb. This

Table 2.2: **Proportion of ng42M calls reciprocally overlapped by GADA T9 calls**

Frequency	(1kb,10kb]	(10kb,20kb]	(20kb,100kb]	(100kb,500kb]	(500kb,+∞]	All Classes
(0,1]	2.35%	9.38%	19.17%	19.15%	10.00%	5.63%
(1,5%]	1.67%	11.22%	7.30%	9.09%	0.00%	3.57%
(5%,10%]	1.86%	13.25%	5.08%	4.85%	50.00%	3.67%
(10%,100%]	0.78%	5.14%	5.09%	7.26%	16.67%	2.30%
All Classes	1.10%	6.77%	6.26%	7.63%	15.63%	2.83%

Table 2.3: **Proportion of GADA T9 calls reciprocally overlapped by ng42M calls**

Frequency	(1kb,10kb]	(10kb,20kb]	(20kb,100kb]	(100kb,500kb]	(500kb,+∞]	All Classes
(0,1]	44.44%	50.00%	66.67%	77.78%	50.00%	54.64%
(1,1%]	69.57%	70.59%	42.86%	58.33%	0.00%	57.60%
(1%,5%]	45.83%	52.94%	44.83%	45.65%	0.00%	45.64%
(5%,10%]	49.25%	51.43%	48.39%	40.00%	33.33%	47.13%
(10%,100%]	31.21%	55.74%	56.61%	51.79%	4.35%	44.82%
All Classes	42.86%	55.06%	52.22%	49.37%	10.26%	47.51%

Table 2.4: **Proportion of ng42M calls reciprocally overlapped by GADA T10 calls**

Frequency	(1kb,10kb]	(10kb,20kb]	(20kb,100kb]	(100kb,500kb]	(500kb,+∞]	All Classes
(0,1]	1.73%	7.59%	17.92%	19.15%	10.00%	4.82%
(1,5%]	1.11%	10.20%	6.74%	9.09%	0.00%	3.01%
(5%,10%]	1.54%	12.05%	3.73%	3.88%	0.00%	3.00%
(10%,100%]	0.63%	4.21%	4.77%	7.04%	16.67%	2.04%
All Classes	0.86%	5.70%	5.76%	7.36%	12.50%	2.46%

Table 2.5: **Proportion of GADA T10 calls reciprocally overlapped by ng42M calls**

Frequency	(1kb,10kb]	(10kb,20kb]	(20kb,100kb]	(100kb,500kb]	(500kb,+∞]	All Classes
(0,1]	59.26%	30.00%	65.52%	85.71%	50.00%	60.00%
(1,1%]	58.33%	63.16%	48.72%	63.64%	0.00%	55.66%
(1%,5%]	51.85%	51.16%	44.62%	50.00%	12.50%	48.12%
(5%,10%]	42.11%	53.57%	40.00%	40.48%	0.00%	41.72%
(10%,100%]	33.09%	56.86%	62.05%	54.35%	5.26%	48.56%
All Classes	43.92%	53.64%	55.32%	51.35%	8.33%	48.95%

Table 2.6: **Proportion of ng42M calls reciprocally overlapped by Birdseye LOD5 calls**

Frequency	(1kb,10kb]	(10kb,20kb]	(20kb,100kb]	(100kb,500kb]	(500kb,+∞]	All Classes
(0,1]	4.69%	18.30%	26.67%	14.89%	10.00%	9.19%
(1,5%]	3.90%	13.27%	9.55%	9.09%	0.00%	5.69%
(5%,10%]	2.67%	19.88%	5.42%	7.77%	0.00%	5.00%
(10%,100%]	1.31%	7.40%	7.45%	9.90%	16.67%	3.41%
All Classes	1.96%	10.27%	8.82%	9.87%	12.50%	4.27%

was likely caused by the resolution difference between Nimblegen 42M arrays and Affymetrix 6.0 arrays, which has two implications: (*i*) Nimblegen 42M arrays have much better power to detect small to middle size CNVs, (*ii*) One large Affy6 call may be called as multiple smaller CNVs in ng42M. Actually if using 'overlap ng42M $>$50%' as the criteria instead of 'reciprocal overlap $>$50%' (see section 2.2.10), specificity also increased as call size increased and reached close to 100% above 500kb (data not shown). Breaking down by CNV frequency, both sensitivity and specificity decreased as frequency increased. This might reflect the enrichment of common CNVs in duplicated regions of the genome and the impaired performance of CNV discovery algorithms in such regions, and suggests a need for different strategy for calling common CNVs. Overall, Birdseye LOD10 call sets achieved the high-

Table 2.7: **Proportion of Birdseye LOD5 calls reciprocally overlapped by ng42M calls**

Frequency	(1kb,10kb]	(10kb,20kb]	(20kb,100kb]	(100kb,500kb]	(500kb,+∞]	All Classes
(0,1]	62.00%	69.57%	51.61%	66.67%	0.00%	60.19%
(1,1%]	72.41%	61.54%	65.79%	66.67%	50.00%	67.65%
(1%,5%]	60.29%	79.63%	47.44%	39.13%	0.00%	56.96%
(5%,10%]	45.88%	69.57%	36.96%	40.63%	33.33%	45.50%
(10%,100%]	43.88%	51.32%	53.20%	48.23%	0.00%	48.91%
All Classes	52.06%	60.79%	51.63%	46.58%	18.18%	52.41%

Table 2.8: **Proportion of ng42M calls reciprocally overlapped by Birdseye LOD10 calls**

Frequency	(1kb,10kb]	(10kb,20kb]	(20kb,100kb]	(100kb,500kb]	(500kb,+∞]	All Classes
(0,1]	2.83%	12.95%	22.92%	14.89%	10.00%	6.75%
(1,5%]	1.89%	12.24%	8.43%	7.27%	0.00%	3.90%
(5%,10%]	1.46%	15.06%	5.08%	6.80%	0.00%	3.61%
(10%,100%]	0.97%	4.21%	6.29%	8.58%	11.11%	2.62%
All Classes	1.27%	6.77%	7.54%	8.62%	9.38%	3.20%

est specificity (55.59%) and the second highest sensitivity (3.20%) of the three. It was particularly better in calling smaller events (1k to 20kb). It had lower specificity than GADA call sets in middle to large size ranges, but it might be affected more severely by the array difference discussed above as having a much larger median call size.

I investigated how sensitivity and specificity changes as a function of call filters. For Birdseye calls, LOD score was a natural quality filter. As APT and GADA did not compute a per call confidence/quality score, a heuristic formula (see section 2.2.11) previously shown to be monotonically related to false positive rate [38] was used. To account for the fact that ng42M calls were relative to a certain reference individual, sensitivity and specificity was calculated based on both direct comparison of Affy6 CNV calls to ng42M CNV calls in the same individual and comparison of

Table 2.9: **Proportion of Birdseye LOD10 calls reciprocally overlapped by ng42M calls**

Frequency	(1kb,10kb]	(10kb,20kb]	(20kb,100kb]	(100kb,500kb]	(500kb,+∞]	All Classes
(0,1]	90.48%	70.59%	62.96%	66.67%	0.00%	72.46%
(1,1%]	73.33%	61.11%	75.86%	66.67%	50.00%	70.33%
(1%,5%]	67.16%	87.80%	55.74%	47.37%	0.00%	63.64%
(5%,10%]	54.00%	91.67%	44.44%	42.86%	0.00%	52.10%
(10%,100%]	49.74%	44.05%	54.15%	46.46%	0.00%	49.85%
All Classes	57.98%	62.21%	55.92%	47.60%	10.00%	55.59%

Affy6 CNV calls to ng42M CNVEs. Birdseye calling plus LOD score filtering outperformed other call sets from other algorithms filtered using the heuristic score under most stringency thresholds by yielding more calls while achieving higher specificity (Figure 2.3).

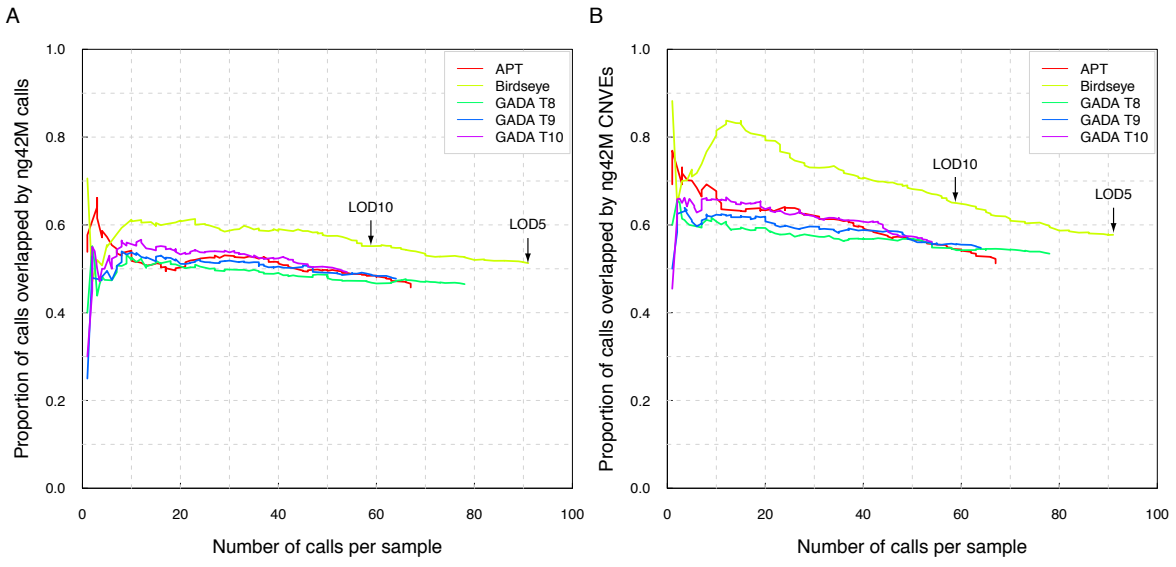

Figure 2.3: Sensitivity, measured by median of number of calls per sample, versus specificity, measured by proportion of calls reciprocally overlapped by ng42M CNVs in the same sample (A) or by ng42M CNVEs (B), under shifting call filters. LOD score (for Birdseye calls) and number of probes times absolute $\log_2$ ratio (for APT and GADA calls) thresholds increase from right to left.

Since calculation of specificity and sensitivity was based on the definition of overlap, I examined if the superior performance of Birdseye was independent of overlap threshold. By fixing the call filter and shifting the reciprocal overlap threshold used to define a test call as being present in the gold standard dataset, a series of specificity value were calculated. Again, Birdseye call sets out-performed other call sets and Birdseye LOD10 had higher specificity than Birdseye LOD5, as expected (Figure 2.4).

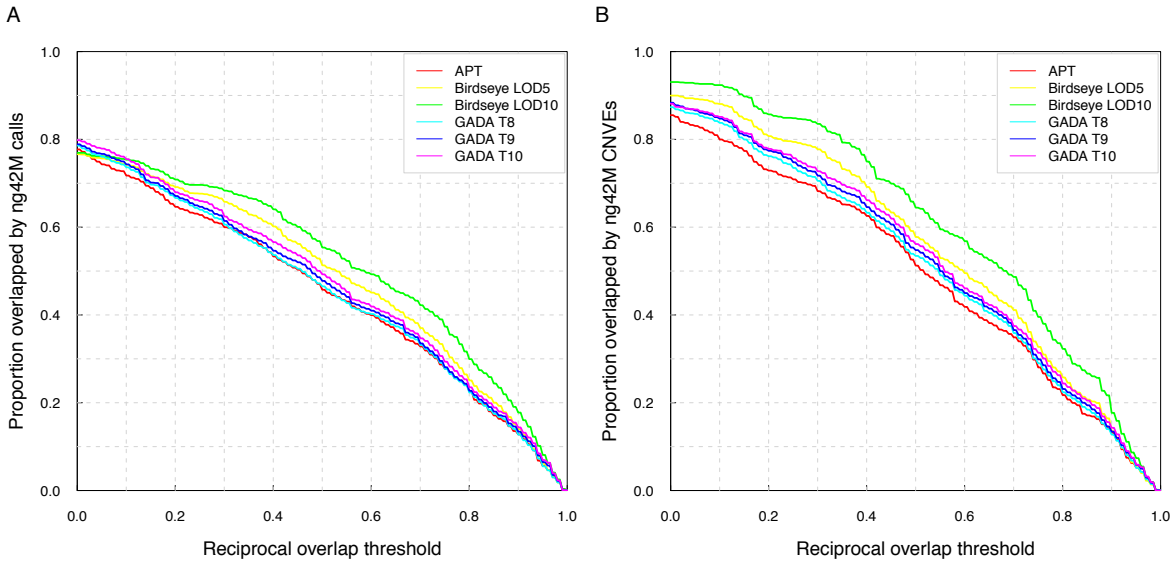

Figure 2.4: Specificity, measured by proportion of calls reciprocally overlapped by ng42M CNVs in the same sample (A) or by ng42M CNVEs (B), as a function of reciprocal overlapping threshold.

Based on the above comparisons, Birdsuite was chosen as the core CNV discovery program around which the production pipeline was built.

2.3.2 Implementing a CNV discovery and QC pipeline for Affy6 data

I developed a robust CNV calling and quality control pipeline for Affymetrix 6.0 data around Birdsuite. The pipeline is able to process thousands of samples automatically, providing robust CNV calls ready for downstream analysis and visualization for manual examination of calling quality. Below I have used the WTCCC2 control dataset as an example when demonstrating certain features of the pipeline.

2.3.2.1 Pre-calling QC

Defects in array experiment can sometimes be visually apparent when simply looking at the scanned image and could lead to the exclusion of the array before entering the CNV discovery process. As the actual scanned images are not available for many array experiments, they were regenerated from the .CEL files (see section 2.2.1). Those with defects such as contamination (Figure 2.5A) and global low-hybridization (Figure 2.5B) can be easily distinguished from those with scanned images of typically normal experiments (Figure 2.5D). Usually, small-scale contamination has little impact on the overall quality of data, but abnormally low hybridization often causes increased noise level. Samples with such defects could also be identified by other QC metric at later stages. Therefore, except for some very rare cases (Figure 2.5C), the role of scanned images generated at this step was mainly to aid the investigation of the cause of low quality data rather than dropping samples before CNV calling.

2.3.2.2 Pre-processing and CNV calling

CNVs were called by plate using Birdsuite with default parameters. The normalized and summarized probe set intensities produced by apt-probeset-summarize, which was called by Birdsuite to handle pre-processing, were stored in the form of in-file database (see section 2.2.6). A copy of the intensities was transformed to log ratios and were used to calculate average log ratio for each CNV call and log-ratio-related

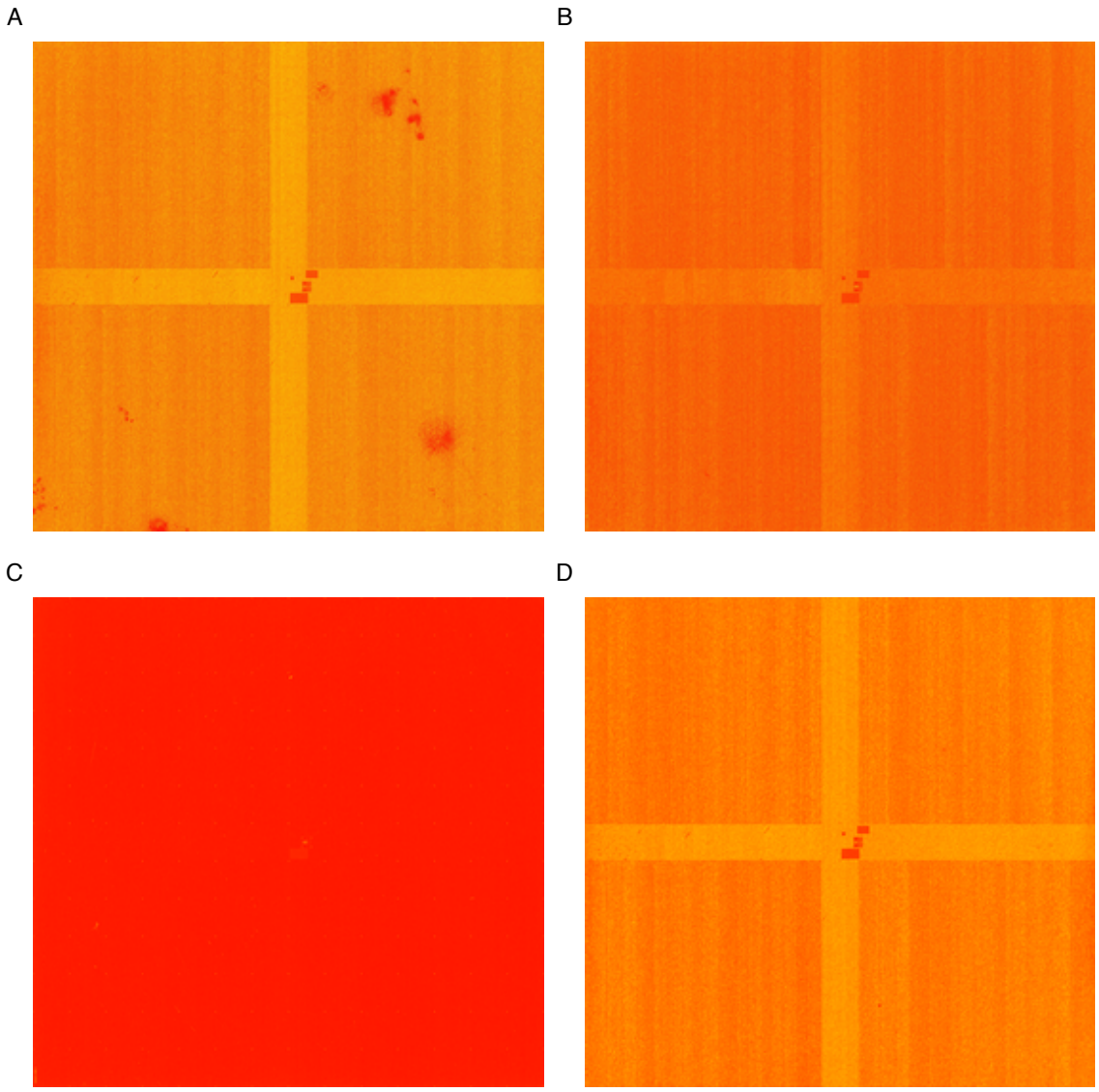

Figure 2.5: Regenerated scanned images of four samples. An Affy6 chip contains close to 7 million probes organized in a 2572 $\times$ 2680 matrix, each represented as a dot in heat colors. Brightness is proportional to $\log_2$ intensity. The four scanned images are examples of contamination (A, scattered abnormal low intensity regions in top right, bottom right and bottom left zone), global low-hybridization (B, globally darker color and blurred border between the central cross region and the rest of the chip), failed experiment (C, no hybridization signal at all) and normal scanned image (D, bright and clear cross region with the rest of chip being relatively homogeneous).

sample QC statistics for each sample. CNVs called from all plates were pooled and stored in CNV call format.

2.3.2.3 CNV call QC

A number of summary statistics were calculated and visualized to aid the decision of filter parameters (Figure 2.6). Considering that Birdsuite does not yet correctly segment Y chromosome and CNV calling in X chromosome is problematic due to the presence of pseudoautosomal regions and the difference in neutral copy number between male and female, and in order to remove the lower end tail in the distribution of call size, number of probes and probe density, I set up the following criteria to filter CNV calls:

1. Autosomal
2. LOD score ≥ 10
3. Number of probes ≥ 5
4. Size $\geq$ 1kb
5. Probe density ≥ 1 per 10kb

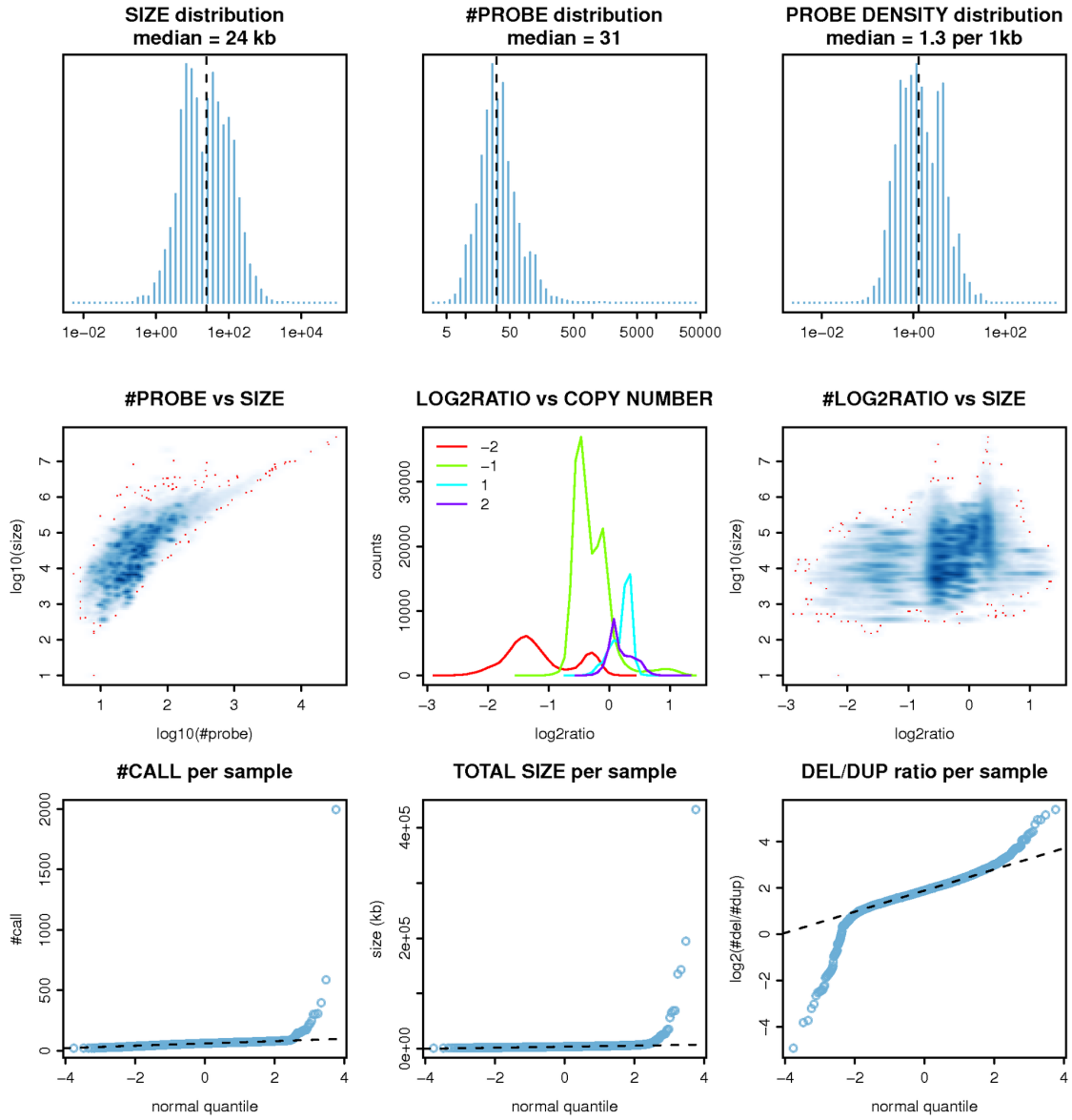

Figure 2.6: Summary statistics of the call set before call QC.

2.3.2.4 Sample QC

Even after the above filters, the number of CNVs called in some samples were a few orders of magnitude greater than most of the other samples. Obviously, this variation could not be explained solely by natural variation in the number of CNVs carried by an individual, but were more likely technical artifacts. There can be two types of such artifacts. The first type is caused by differential sensitivity, wherein specificity of CNV calling is good and similar across samples, but due to some

samples being noisier than others, fewer CNVs can be called with a level of confidence that reaches a pre-defined common threshold. The second type is more severe, wherein the data quality of some samples is so poor that CNV calling program starts to produce large number of false calls. This was observed when running samples with strong waviness as indicated by having high spatial auto-correlation through the GADA test pipeline without correction. The number of apparently false CNVs was effectively reduced after correcting for spatial auto-correlation, suggesting waviness was indeed the cause of such artifact (Figure 2.7). As Birdseye works on probe set intensities rather than log ratios, such correction could not be performed, which might explain the excessive CNV calls. To account for both types of artifact, I used a linear function to model the negative correlation between the number of calls per sample and the sample's median absolute deviation (MAD) of log ratios, a measure of the level of noise in the data, wherein the parameters of the linear model were estimated using samples with a MAD$<$0.3 and a SAC in the bottom 90% in order to exclude the influence of samples with extreme level of noise or spatial auto-correlation. Samples which after correction were more than four MADs from the fitted linear model were removed (Figure 2.8).

In addition to the number of calls per sample, deletion-to-duplication ratio (DDR) should also be relatively stable across samples given the same CNV discovery algorithm and a reasonably large number of CNVs called per sample. Indeed, most samples had a DDR between 1 and 16 with a median at about 4 (Figure 2.9). The majority of samples that fell outside this range had a DDR below the lower bound and many of them also had high SAC and coincided with those having excessive CNV calls. This indicates the abnormally high number of calls per sample and low DDR was predominately driven by over-calling of false duplications in samples with strong spatial correlation. In practice, the median and MAD of the distribution of log-transformed DDR were calculated and samples falling more than four MADs from the median were removed (Figure 2.9).

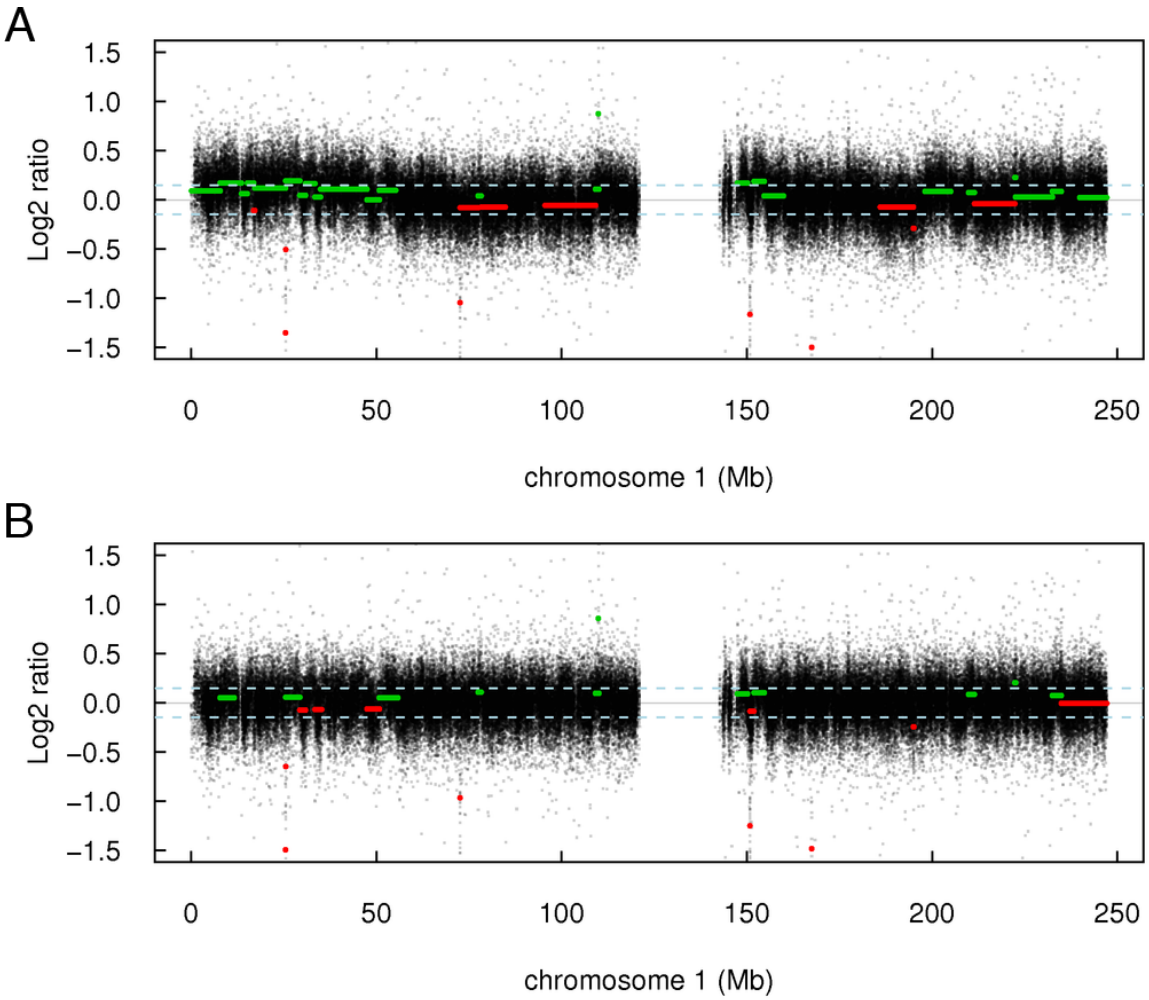

Figure 2.7: Example of over-calling due to spatial waviness along the chromosome. Black dots are $\log_2$ ratios across chromosome 1 of a sample with a SAC of 0.78 (top 0.15%) before (A) and after (B) correction for spatial auto-correlation. Horizontal segments are deletions (red) and duplications (green) called by GADA. The average $\log_2$ ratios of the CNV calls are represented by the horizontal position of the segments. Those located between -0.15 and 0.15 (light blue dashed lines) will be filtered out.

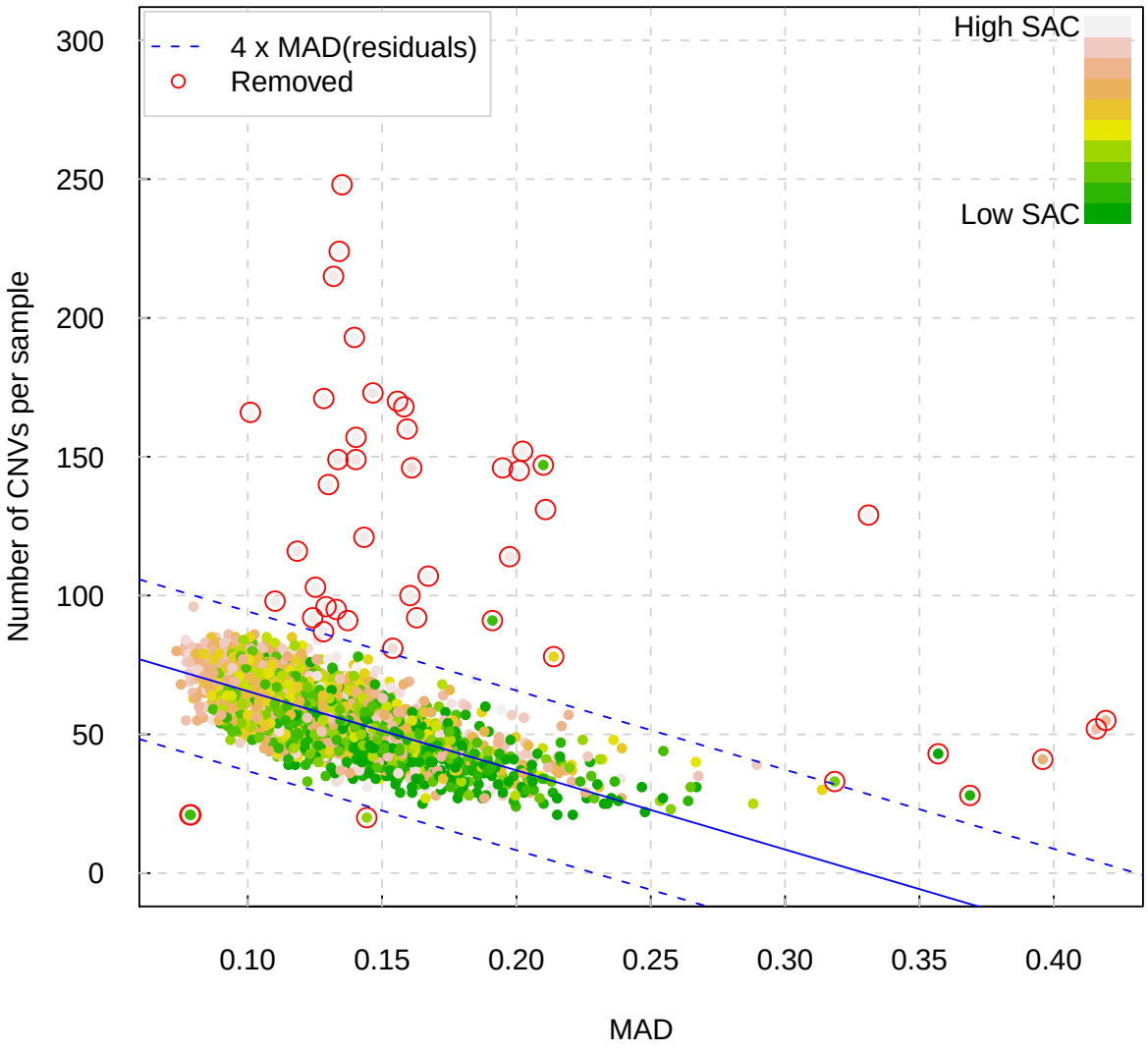

Figure 2.8: Number of CNV calls per sample as a function of the sample's level of noise. Each solid colored dot represents a sample. The sample's level of spatial auto-correlation is coded by terrain color, where the more greenish the lower the level of spatial auto-correlation. The blue solid line denotes the fitted linear model. The blue dashed lines represent four times the MAD of the residuals away from the fitted line. Dots encircled by red are samples to be removed for falling outside the region bordered by the blue dashed lines.

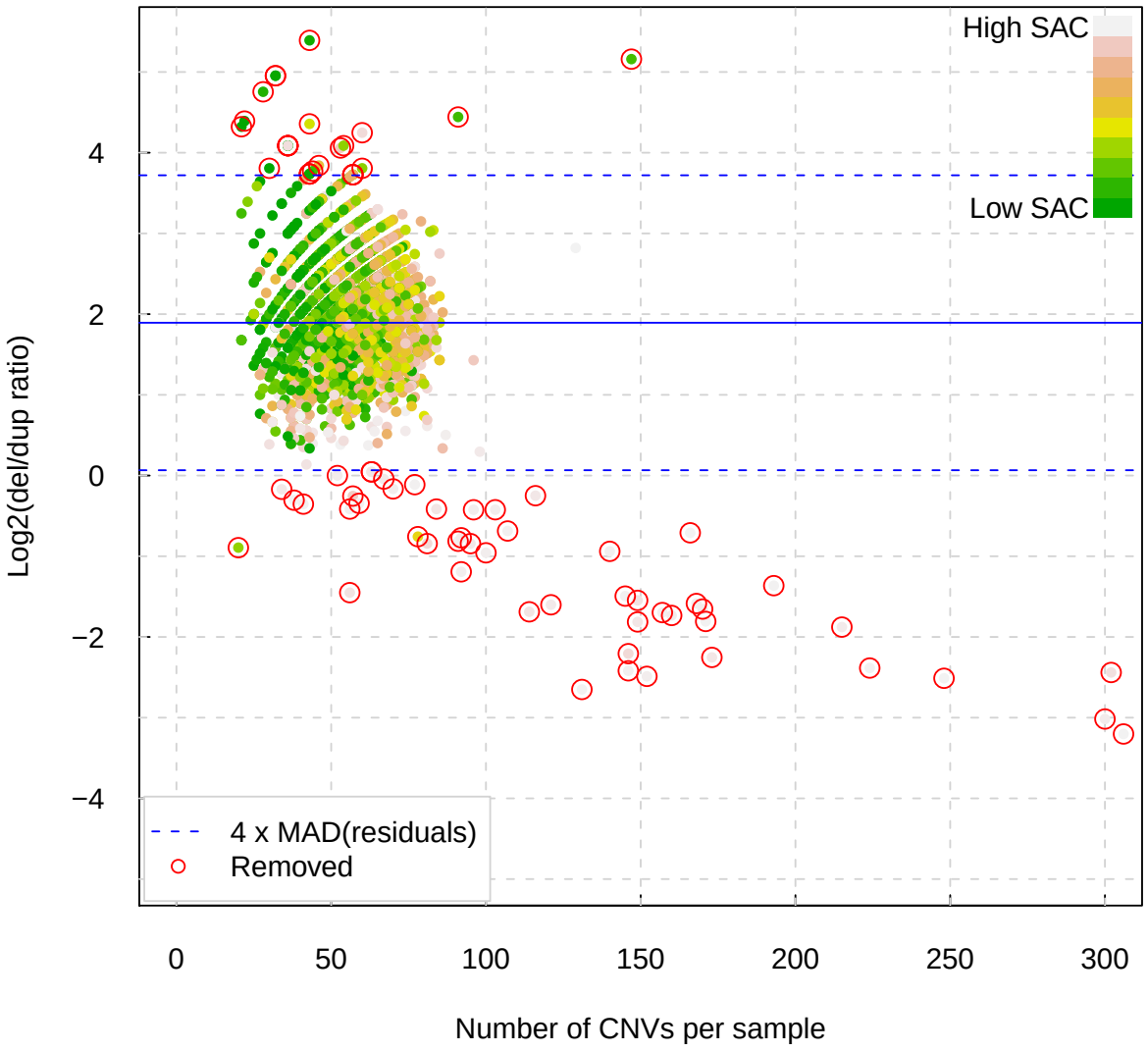

Figure 2.9: Distribution of deletion-to-duplication ratio. Each solid colored dot represents a sample. The sample's level of spatial auto-correlation is coded by terrain color, where the more greenish the lower the level of spatial auto-correlation. The blue solid line denotes the median of $\log_2$ deletion-to-duplication ratio. The blue dashed lines represent four times the MAD of the residuals away from the median. Dots encircled by red are samples to be removed for falling outside the region bordered by the blue dashed lines.

2.3.2.5 Merge split CNV calls

Birdseye sometimes incorrectly split large CNVs into multiple smaller calls due to just a few probes in the middle that did not meet the expected level of dosage-responsiveness (Figure 2.10). I added an *ad hoc* step to merge these split calls based on the number of probes between adjacent calls, the ratio of the number of probes between adjacent calls to the number of probes in the merged call, the probe density between adjacent calls and the absolute difference in log ratio between adjacent calls (see section 2.2.8). My selection of merging parameter values was guided by the distribution of the above metrics (Figure 2.11) and visual inspection of the merged calls.

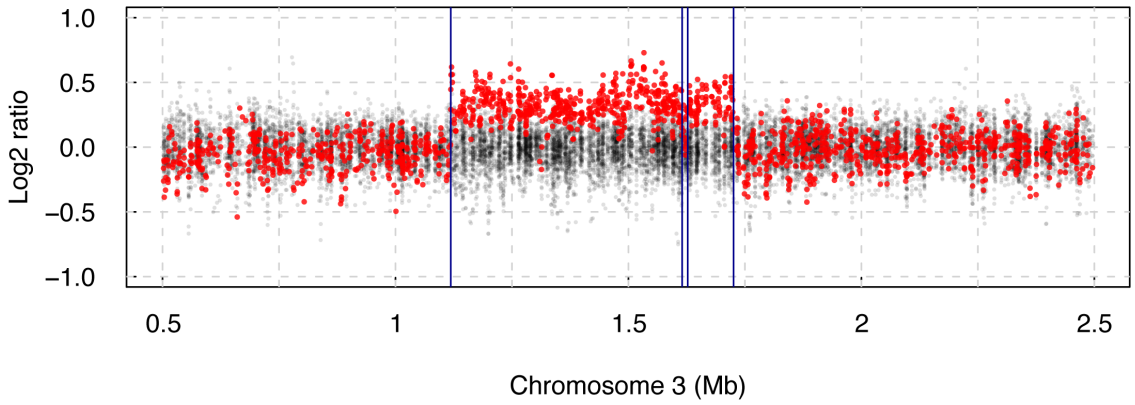

Figure 2.10: Birdseye split a duplication of 620kb into two duplication calls of 500kb and 110kb, respectively. The sample carrying the duplication is highlighted in red. Other samples in the same plate are in black. Blue vertical lines denote the boundaries of the two duplication calls.

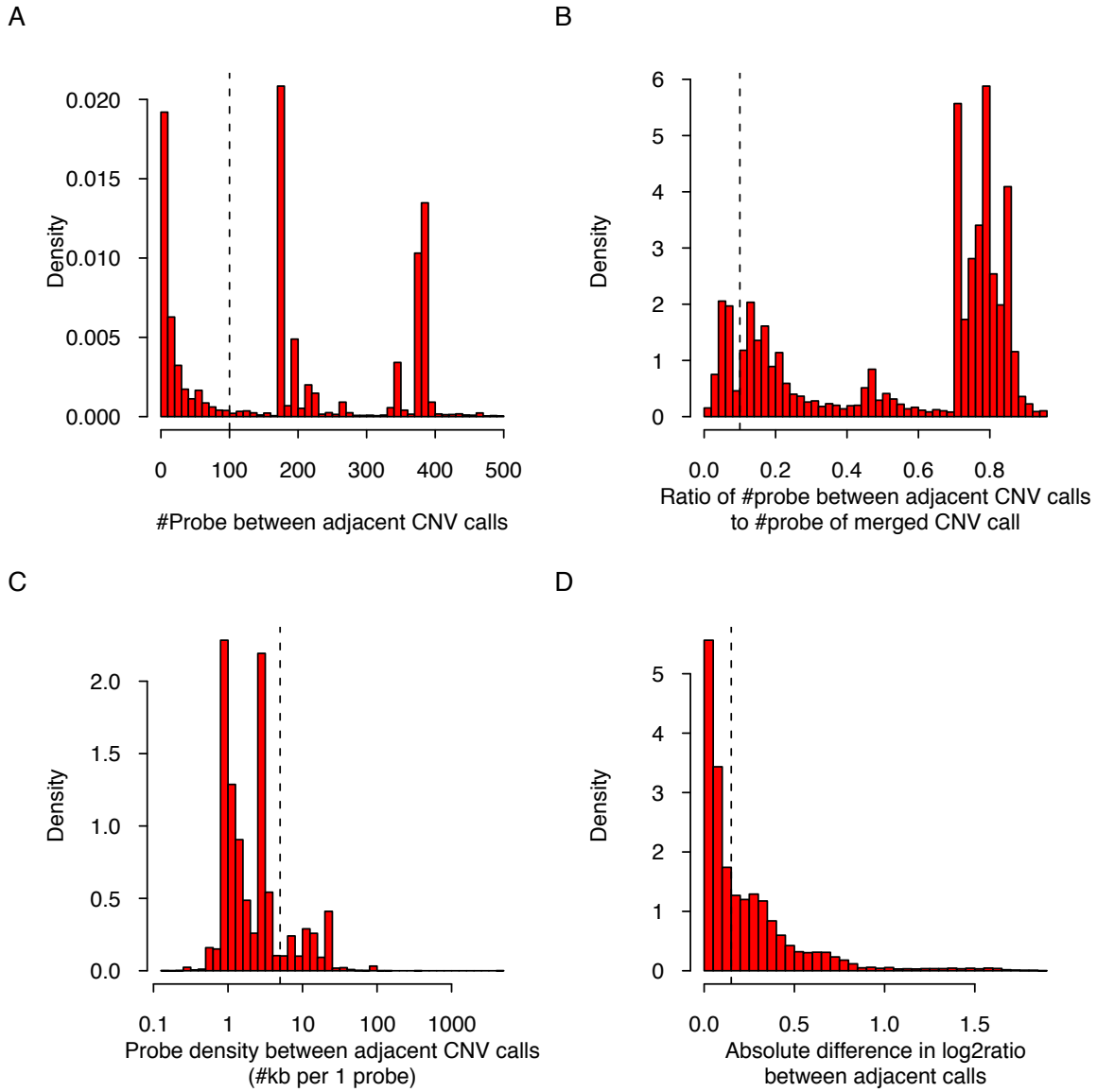

Figure 2.11: The distribution of variables used as metric for deciding if adjacent calls should be merged. Vertical dashed lines mark the thresholds.

2.3.2.6 Cluster CNVE and calculate CNVE frequency

I observed frequently that CNV calls discovered in one sample had extensive overlap with CNV calls in multiple other samples, which likely indicate these variants were identical and probably result from a single ancestral mutation event. Under such a scenario, any slight differences in location were probably just technical fluctuations in the precision of CNV discovery. Even if two overlapping CNV calls orig-

inated independently in different individuals and had real slight differences in their locations, operationally treating them as a single event was reckoned reasonable in most analyses considering the highly similar genomic content they encompass and the utility of knowing the frequency of a CNV of a particular genomic interval. I used a clustering-like algorithm to merge such CNVs into CNVEs (see section 2.2.9) and the frequency of a CNVE was calculated as the number of individuals carrying this CNVE divided by the sample size. This call frequency is not the same as an allele frequency, but is nevertheless useful in downstream analyses to distinguish between common and rarer CNVs.

2.3.3 Application of the pipeline to process Affy6 datasets

I applied the above CNV discovery pipeline to the following cohorts genotyped using Affy6:

1. 5,989 UK individuals recruited as common controls in the Wellcome Trust Case Control Consortium 2 project (referred to as WTCCC2).

2. 1,442 American individuals of European ancestry recruited as controls for the GAIN study of Schizophrenia and Bipolar disease (referred to as GAIN_EA, Genetic Association Information Network, European Ancestry)

3. 226 prenatal samples with major ultrasound abnormalities or multiple soft markers detected by standard two dimensional ultrasonography, (referred to as AFD, Abnormal Fetal Development)

4. 334 UK patients with sever-early-onset obesity, half of which also had developmental delay (referred to as SCOOP1, Severe Childhood-Onset Obesity Project 1)

5. 1,386 UK patients with severe early-onset obesity (referred to as SCOOP2)

The biological interpretation of the CNVs identified in these cohorts is described in other chapters in this thesis. Here I focus on the performance of the CNV discovery pipeline across a range of different datasets, generated in different laboratories. I compared the QC metrics and CNV statistics of these cohorts, examined the reproducibility of CNV discovery using this CNV calling pipeline and investigated if commonly adopted QC metrics for SNP genotyping are also appropriate for CNV QC.

2.3.3.1 Comparing QC and CNV statistics

I first examined the distribution of level of noise and spatial autocorrelation in the five datasets. Small but statistically significant differences in the distribution of the level of noise were observed both between control cohorts and between controls

and cases (Figure 2.12A). The level of noise (as measured by the MAD of log ratios) was lower in WTCCC2 as compared to GAIN_EA ($p = 2.2\times10^{-47}$, two-sided Mann-Whitney U test, same for the following), AFD ($p = 8.3\times10^{-9}$), SCOOP1 ($p = 8.6\times10^{-13}$) and SCOOP2 ($p = 1.1\times10^{-5}$). These differences largely explained the differences in the distribution of number of calls per sample among the different cohorts (Figure 2.12B, r = -0.88, p = 0.04). Significant differences in the distribution of spatial autocorrelation were also observed. SCOOP1 and SCOOP2 samples had significantly greater median spatial autocorrelation ($p = 6.3\times10^{-60}$ and $p = 2.0\times10^{-100}$, respectively, as compared with WTCCC2) and more samples with very high spatial autocorrelation than the rest of the cohorts (Figure 2.12C). This explained their lower sample QC pass rate (Figure 2.12D, r = -0.93, p = 0.02).

Table 2.10: **Summaries statistics of CNV call sets**

Cohort	#Sample pass QC	Median #call per sample	Median call size (kb)	Deletion-to-duplication ratio	#CNVE	Median CNVE size (kb)	%Singleton CNVE	Average plate size
WTCCC2	5897	58	23.6	3.73	12295	37.9	62.8	84
GAIN_EA	1419	49	27.0	3.83	4493	42.9	63.4	85
AFD	224	50	22.3	5.13	1432	33.1	58.9	38
SCOOP1	292	55	23.5	4.09	2173	32.7	61.7	67
SCOOP2	1289	56	23.3	3.91	5277	37.9	64.5	87

Next I compared the summary statistics of the final filtered call set of the different cohorts (Table 2.10). As discussed above, GAIN_EA and AFD produced fewer calls per sample due to having noisier intensities ($\log_2$ ratios). GAIN_EA had larger CNVs and CNVEs, possibly due to lower sensitivity to smaller events, but there could be other contributing factors. The differences in the proportion of CNVEs seen only in one sample (singletons) might be partly explained by differences in the sizes of the cohort and possibly the impact on sensitivity of differences in average batch (plate) sizes (r = 0.97, p = 0.006), since Birdseye receive parameters of the

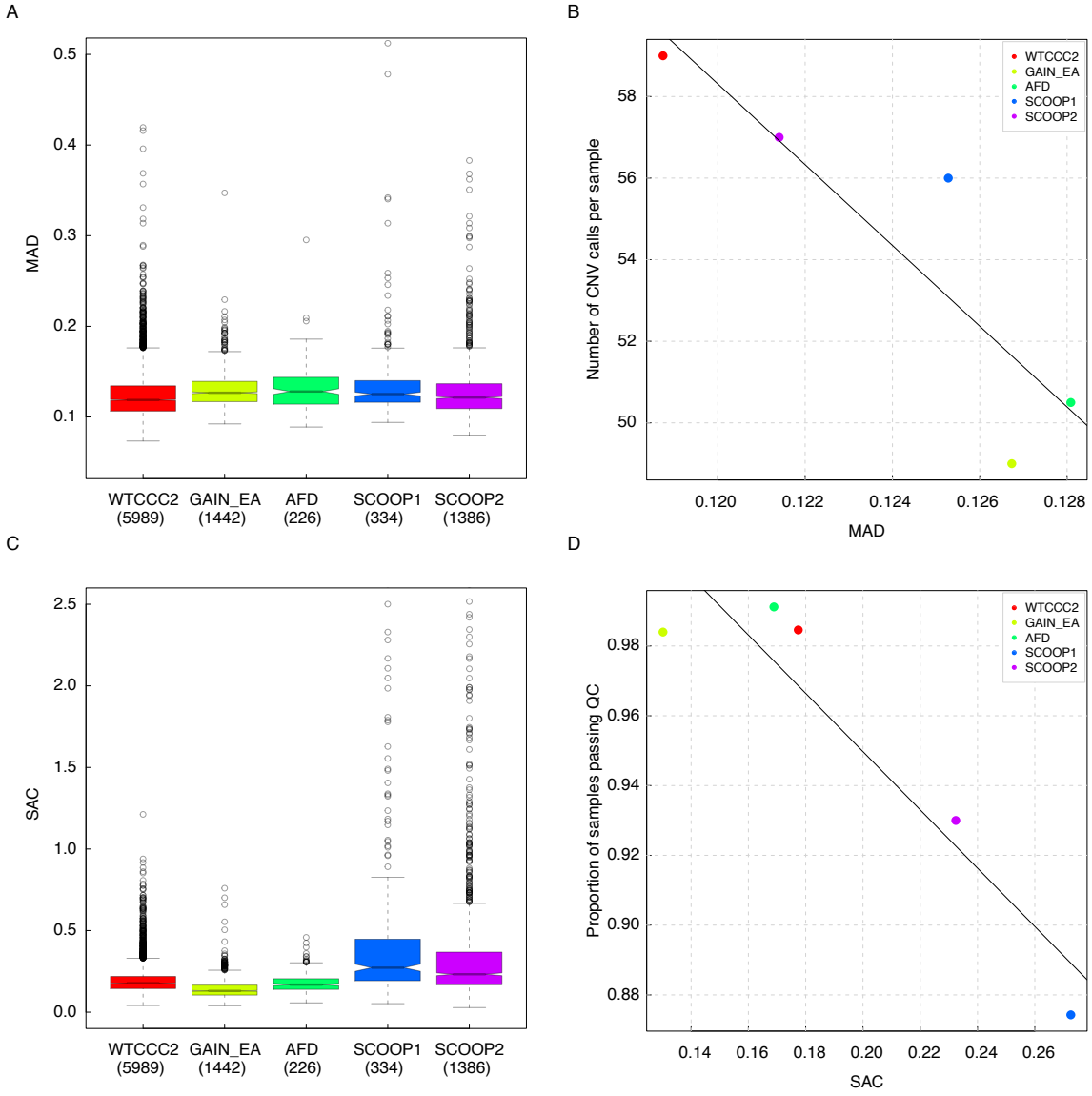

Figure 2.12: Comparison of QC statistics. (A) Distribution of the level of noise. (B) The number of CNV calls per sample as a function of the level of noise. The points represent the median value for each cohort. (C) Distribution of spatial autocorrelation. (D) Sample QC pass rate as a function of the level of spatial autocorrelation. The x value of each point is the median SAC for each cohorts.

emission probability distribution from Canary and Canary could overestimate the variance of the intensity distribution of the neutral copy state when given a smaller number of samples, which would lead to under-calling of singletons. The disease cohorts (AFD, SCOOP1 and SCOOP2) had greater deletion-to-duplication ratios,

which might reflect ture biological differences, but more likely is due to technical biases, as duplications become more difficult to call than deletions with noisier data and smaller plate sizes.

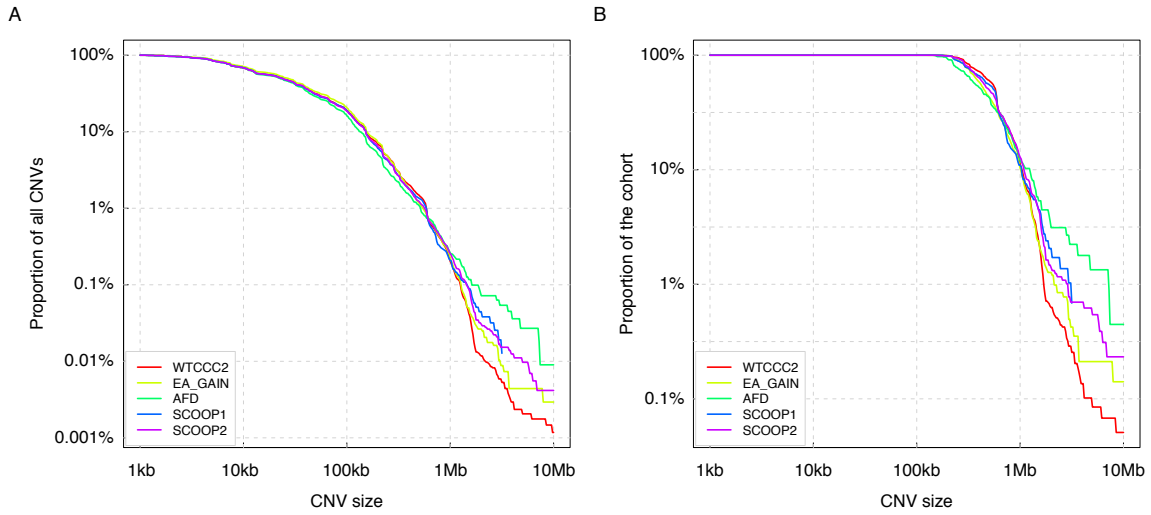

Figure 2.13: Proportion of large CNVs relative to all CNVs as a function of size threshold (A) and proportion of the cohort carrying large CNVs as a function of size threshold (B). Both CNV sizes and proportions are in log scale.

Finally, I investigated if there is difference in the distribution of large CNV calls in the call sets. As the calling of large CNVs should be least affected by technical issues, this could provide insights into the biological characteristics of the cohorts. For CNVs exceeding a certain size threshold, I calculated their proportion relative to all CNVs and the proportion of the cohort carrying such CNVs. The proportions remained relatively similar until the threshold reached 1Mb, beyond which disease cohorts (AFD, SCOOP1 and SCOOP2) had both greater proportion of large CNVs and greater proportion of individuals carrying such CNVs (Figure 2.13).

2.3.3.2 Reproducibility of CNV discovery using Affy6 plus the pipeline

There were 55 SCOOP1 patients genotyped for a second time using Affy6 as part of SCOOP2 (46 passed sample QC both times), which provided an opportunity to investigate the reproducibility of CNV discovery using the CNV discovery pipeline I developed. Samples of 46 of those patients passed QC in both datasets. For each

individual, I defined a CNV called in one dataset 'reproduced' if it reciprocally overlapped >50% with a CNV called in the same individual in the other dataset. Replicate rate was defined as:

$$\frac{N_{\text{reproduced,SCOOP1}} + N_{\text{reproduced,SCOOP2}}}{N_{\text{SCOOP1}} + N_{\text{SCOOP2}}}$$

On average, a replicate rate of 76.8% was achieved. As expected, due to differences in sensitivity and specificity, the replicate rate was much higher for deletions than duplications (Table 2.11). I further interrogated if the concordance ('replicate rate') between CNV sets called in samples from the same individual was higher than that between CNV sets called in samples from different individuals. To do this, for each of the 46 SCOOP1 samples, I calculated replicate rates with 100 randomly chosen SCOOP2 samples. This verified that the observed level of reproducibility between samples from the same individual was not a mere coincidence that could be achieved by pairing randomly chosen samples (Figure 2.14).

Table 2.11: **Replicate rate of CNV discovery using Affy6 and the Birdsuite pipeline**

Type	(1kb,10kb]	(10kb,20kb]	(20kb,100kb]	(100kb,+∞]	All Classes
Duplication	40.00%	63.08%	46.20%	60.97%	54.56%
Deletion	80.08%	76.15%	87.46%	81.25%	82.10%
All Classes	79.19%	74.93%	78.65%	70.82%	76.83%

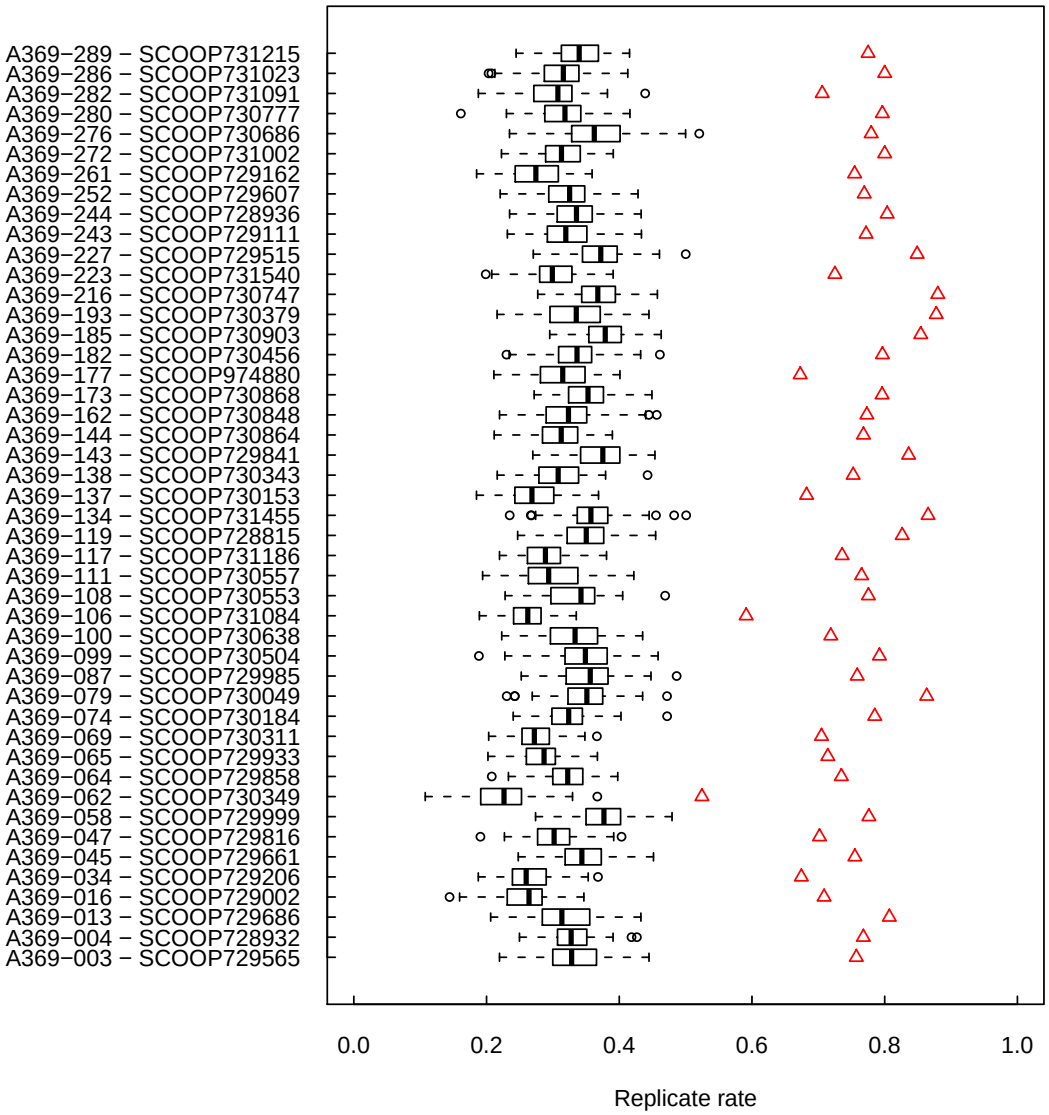

Figure 2.14: Comparing replicate rate between randomly chosen pairs of samples and samples that are true replicates. The distributions of replicate rate between randomly paired samples are presented by boxes whereas replicate rate between true replicates as printed in the labels are presented by red triangles.

2.3.3.3 SNP genotyping QC metrics are not suitable for CNV QC

SNP call rate and the level of heterozygosity are sample QC metrics that are frequently used in SNP GWAS where samples having low SNP call rate or being outliers in the distribution of the level of heterozygosity were removed [13, 39]. I inves-

tigated if these metrics are also appropriate for identifying samples to be removed for CNV analyses. I examined the distribution of SNP call rate and the level of heterozygosity against the two metrics I used for CNV sample QC, the number of calls per sample and deletion-to-duplication ratio (DDR). As shown in (Figure 2.15), the majority of samples having low SNP call rate or being outliers in the distribution of the level of heterozygosity yielded similar number of CNV calls or DDR to samples with high SNP call rate and normal level of heterozygosity. Samples with extreme number of calls and DDR are close to the mode of the distribution of SNP call rate and the level of heterozygosity, implying that filtering samples for downstream CNV analyses by using these SNP-based QC metrics would not be useful.

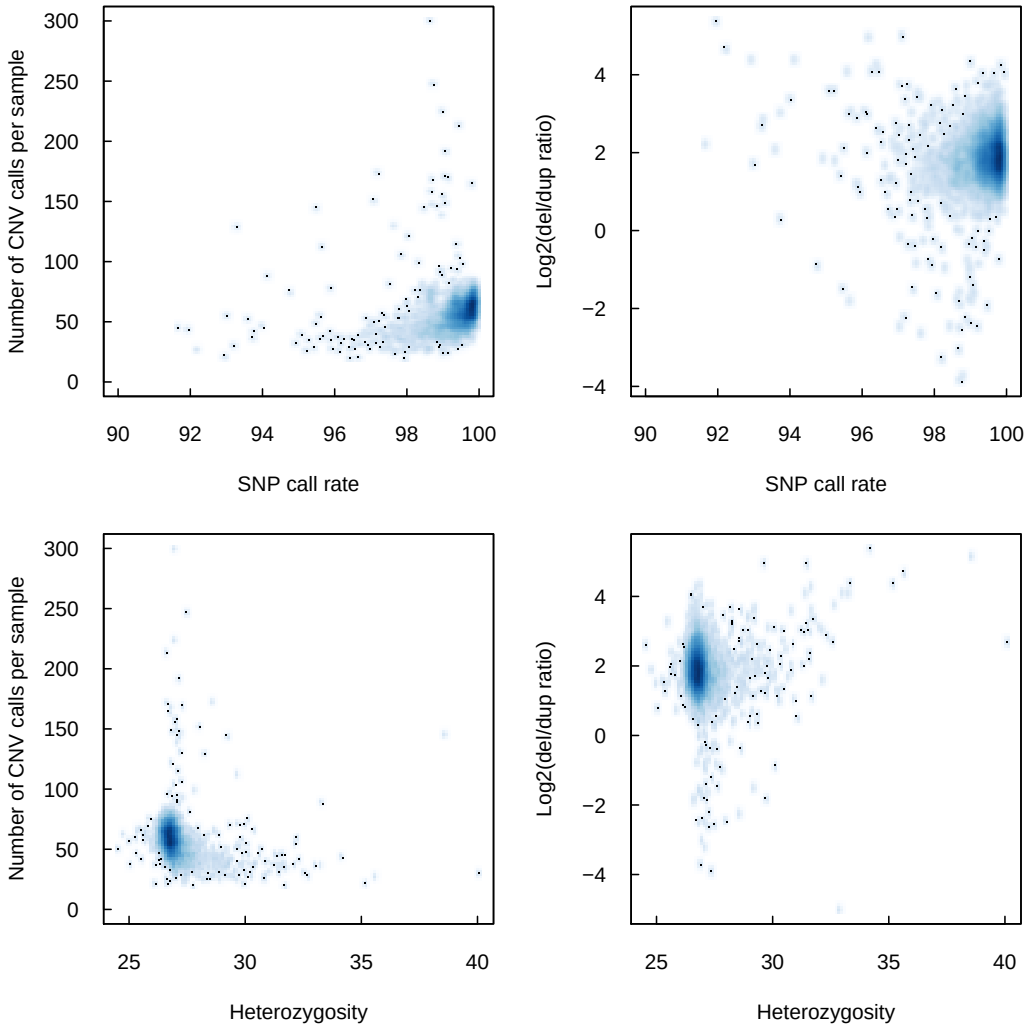

Figure 2.15: Distribution of SNP QC metric against CNV QC metric.

2.4 Discussion

In this chapter, I described the development of a CNV calling pipeline for Affy6 genotype data. I first compared the performance of three CNV calling programs on Affy6 data. Next, I built a production pipeline around the selected program, Birdsuite, which incorporated a number of QC metrics and post-processing procedures. Finally, I applied the pipeline to several Affy6 datasets to produce filtered CNV call sets for further analysis. I have demonstrated that the pipeline I developed generated high quality CNV call sets across a range of different datasets.

2.4.1 Storage of CNV data

The amount of data a single microarray experiment can produce increases linearly with the increase in resolution and coverage of the array. With 6,892,960 oligonucleotide features on the slide, Affy6 yields nearly seven million raw intensity values per experiment, which are summarized to more than 2.7 million intensity values corresponding to nearly one million copy number probe sets and one million SNP probe sets. The summarized and normalized intensities, together with minimal annotations required for CNV calling, including probe set ID, type and genomic location, take up close to 2Gb of disk space for 96 samples if stored in plain text format. Such data are required not only for CNV calling but for QC and various visualizations as well, in which scenario fast access to intensity values of certain samples within a genomic window of interest is needed. Due to hardware limitations, I used a special HDF file to store and manage such data. Although the genomic coordinates were indexed to facilitate fast query, the speed is still affected and limited by disk performance. However, relationships among samples and other annotations associated to probes and samples have to be managed separately. Ideally, all those data should be stored in an efficiently designed database with an index loaded in memory.

2.4.2 Log ratio versus intensity

Birdsuite outperformed GADA and APT for CNV discovery from Affy6 data in the comparisons I performed. Its CNV discovery component, Birdseye, is the only program of the three that works on intensities rather than converted log ratios. In principle, the conversion to log ratio should reduce the variation across different probe sets. Actually, this explains why most general-purpose (multi-platform) CNV discovery programs expect to work with log ratios: for segmentation-based methods, simple piecewise constant function can be used to model log ratio profiles along a chromosome and for HMM-based method, the same parameters for emission model can be used for all probe sets. By comparison, algorithms working with intensities produced from single channel genotyping arrays need special treatment to handle the larger variance across probe sets (*e.g.* Birdseye uses probe-set-specific emission parameters), which often limits their application to other types of arrays. However, calling CNVs from intensities also has advantages over log ratios. First, strictly speaking, log ratios can only indicate comparative loss or gain of copy number relative to a reference rather than an actual genotype. For the APT and GADA test pipeline, log ratio were converted from intensities using a population (all samples in a plate) median as the reference, which could deviate from diploidy in common CNV regions and lead to a more balanced 'deletion' 'duplication' ratio, as shown in Table 2.1, which, if taken at face value, could give a misleading view of the nature of CNVs in an individual's genome. Second, discriminating high copy number states is much harder using log ratios than using intensities especially when the reference's copy number is greater than two.

2.4.3 CNV discovery QC filter parameters

Due to differences in array specification, CNV discovery algorithm and purpose of investigation, there is little consensus in what filters should be applied for CNV discovery. Without an independent and high quality CNV call set in the same individuals, previous studies often have had to rely on simulated data or indirect measures [24, 26, 28]. Rather than using stringent filtering, I have instead used instead fairly permissive filters in the production pipeline, as Birdseye calls with a LOD $\geq$ 10 al-

ready have reasonably high specificity even for smaller events (in the size range of 1kb to 10kb (Table 2.9) or having 5 to 10 probes (data not shown)), as judged in the comparisons with the ng42M call set.

2.4.4 CNV discovery sample QC

The sample QC method I developed for this pipeline is relatively simple yet effective. By quantifying the two primary data quality factors that affect CNV discovery performance, spatial auto-correlation and noise, the method is able to clearly distinguish samples of acceptable quality, in which the number of CNV calls per sample follows an expected inverse correlation with the level of noise, and samples that are apparent outliers to this trend. This pattern of separation has been consistently observed in several Affy6 datasets, and in principle should be applicable to CNV discovery pipelines for other arrays and sequence data as well.

The QC methods can still be improved. In analyses described in later chapters in this thesis, I found that data quality was not equally poor throughout the entire genome and good quality CNV calls at specific loci could still be salvaged in some of the samples that had failed the QC thresholds described here. Therefore, a finer QC procedure that filters by chromosomes rather than by samples might prove useful.

2.4.5 CNV clustering versus joint calling

An *ad hoc* CNV clustering step was deployed at the end of the discovery pipeline to combine CNVs called from each individual that likely correspond to the same mutation event and to calculate a lower bound on the numbers of individuals carrying such events in a population. Based on reciprocal overlap, the generality of this method ensures it can be applied to all CNV call sets produced by various CNV discovery pipelines. However, a better solution would be to statistically model CNVEs and to call CNVs jointly from multiple individuals rather than calling CNVs one individual at a time. As pointed out by Zöllner [40], sharing information across individuals should not only increase the sensitivity of calling common CNVs but also make estimation of the border of CNVE more accurate.

2.4.6 Merging split CNV calls

There have been a number of reports that large CNVs are sometimes incorrectly split by both HMM-based methods [11, 33, 41] and segmentation-based methods [12, 27]. In this pipeline, I introduced a merging step after call QC and sample QC. For some large CNVs, since each individual split calls produced by Birdseye did not meet the call QC thresholds, they could not be caught by the merging step and hence missed from the final call set. An alternative design would be to merge CNV calls prior to any call filtering. However, it would be difficult to derive a LOD score for the merged CNV call to allow it to be filtered along with other, unmerged calls. A neater solution would be to reduce the probability of splitting large CNVs at the discovery stage. HMM-based methods such as Birdseye currently often use distance-aware transition probabilities wherein the likelihood of a probe having a different copy number state from its previous probe increases as the distance to the previous probe increases. These distance-aware transition probabilities are independent of the location of the probe. With the current knowledge of spatial distribution of CNVs across the genome, location-aware transition probabilities could be introduced. With the availability of large amount of Affy6 data, one can calculate a signal-to-noise ratio for every probe and weight probes on their signal-to-noise ratio during CNV calling. Such information has been used in segmentation-based method [42] and can also be incorporated into the Viterbi algorithm for HMM-based methods.

2.4.7 Application of this pipeline

This pipeline has been successfully applied to a number of cohorts genotyped using Affy6, ranging from apparently healthy genomes and patient genomes with subtly different patterns of CNV from controls (see following chapters), and should be applicable to the majority of disease cohorts. However, this pipeline was not designed for CNV discovery in cancer genomes, since (*i*) the emission model parameters of Birdseye were estimated only for the pre-defined states corresponding to copy number of integer 0–5, and (*ii*) spatial auto-correlation and level of noise as measures of data quality only applies to genomes with a limited amount of CNV and the assumption that the number and the deletion-duplication-ratio of CNVs discovered

in one sample should be relatively comparable among individuals does not hold for cancer genomes.