Advanced statistical tools for SNP arrays : signal calibration, copy number estimation and single array genotyping

Rippe, R.C.A.

The SCALA software suite

7

The SCALA Suite for Single SNP chip analysis provides functions to convert raw CEL-files to Rdata objects, one for each chip. Furthermore, given a set of high quality arrays, universal calibration parameters can be computed and applied to new arrays. Genotypes are called on a single chip with a dedicated function. Maps of copy numbers and allelic imbalance are also implemented for single arrays.

7.1 Introduction

SNP (Single Nucleotide Polymorphisms) arrays have two major applications: genotyping of DNA and studying copy number variations (CNV) and allelic imbalance. Here we describe integrated R software called SCALA designed for this purpose. It has a unique combination of properties: it can perform CEL file conversion, genotyping, copy number mapping and signal calibration. After using SCALA no further software is needed. In the remainder of this Introduction we describe the main components of SCALA in more detail.

Genotyping

To estimate all SNP genotypes in a single chip, semi-parametric log-concave mixtures were proposed in Rippe, Eilers & Meulman (2010). One reason is that the latter only works effectively on array sets of reasonable size, in order

This chapter is an adapted version of the submitted article:
Rippe, R.C.A., Eilers, P.H.C. and Meulman, J.J. (2012). SCALA: a software suite for single chip SNP calibration, genotyping and copy number mapping, *submitted for publication*.

to obtain stable estimates. Furthermore, low minor allele frequencies pose additional problems. The above is circumvented when genotypes are called for a single chip. Proposals for small sample sets have been made, also using mixtures (ALCHEMY by Wright et al., 2010), as well as a combination of single and multi-array analysis (MAMS by Xiao et al., 2007). Building on their arguments we have developed an algorithm that performs single array genotyping. It is based on a two-dimensional mixture of log-concave densities (along the lines of Eilers & Borgdorff, 2007), fitted on 2-dimensional histograms (Eilers & Marx, 2007). To estimate a mixture with three smooth components, we use the familiar EM (expectation-maximization) algorithm. Two steps are repeated until convergence: 1) split the counts y into three vectors of pseudo-counts, proportional to the current estimate of the mixture components; 2) apply smoothing to the pseudo-counts. Decent starting estimates for the components are needed. In Rippe et al. (2010) genotype calls from a multi-array method (CRLMM) and from our single-array method (SCALA) are compared to a set of consensus genotypes from HapMap. The number of agreements and differences in terms of homo- and heterozygous calls showed that SCALA can be used to call genotypes efficiently and effectively. Even SNPs that were not genotyped in HapMap can be genotyped with reasonable certainty using a single chip. The above model is implemented in the `SCALA.genotype` function.

Visualization of copy numbers and allelic imbalance

DNA in tumors can show a variety of deviations like allele copy number variation (CNV) and allelic imbalance. SNP arrays provide a fluorescence signal for each allele, both of which are assumed to be proportional to the number of both alleles. Sums ($\log(a+b)$) and ratios ($\log(b/a)$) of these signals can be plotted, on logarithmic scales, versus positions on chromosomes, to give a useful graphical representation (like in DNACopy, 2010; Golden Helix, 2011). These plots can be enhanced in several ways. Here we present an R program, called SCALA.Map, for this purpose. The program offers smoothing of CNV and allelic imbalancesignals.

SNP signal intensity calibration

SNP fluorescence signals are not perfect: they contain "noise". This noise appears not to be random, implying that it can be modeled in order to correct for it and so calibrate the intensity signals. A remarkable and useful property of fluorescence signals from all types of platforms is that they contain a specific structure. First, there is the (trivial) difference between arrays, which most readers are familiar with. However, a similar pattern also holds for individual SNPs over sets of arrays, and this is what we exploit here. Given a set of called genotypes for the current array, one can obtain calibration parameters for arrays and SNPs. These parameters need to be estimated only once for a given chip type, based on a set of (high quality) arrays. Once a set of calibration parameters has been estimated, it can be used to calibrate the signals in any new individual array, without the need of knowing the genotypes. We estimate α using a set of high quality samples of normal tissue. Estimation of these parameters is implemented in the function `SCALA.calibrate`. The calibration is very effective in copy number mapping, but it can also be used in genotyping. The latter is only offered as an experimental function; assessment has yet to be performed.

Functions and a graphical user interface

The modules for file conversion, genotyping and calibration are accessed via regular function calls. Copy number mapping is done through a custom graphical user interface, which controls the plot settings as well as export options.

7.2 Functions and implementation

In this section we describe the main function in the software suite. We start with CEL files conversion, then discuss calibration, genotyping, and finish with a graphical interface for copy number estimation. Supporting data files and subfunctions are placed in the relative folder locations `'../Maps'` and

'../Calibration' (for file conversion), and '../Support Files' (for genotyping and mapping).

File conversion: SCALA.convert

The software is built around an object of class SCALA, which is a conversion of a raw CEL file to aggregated fluorescence signals for each allele. The conversion function is specific for each chip type, but the result is generic. Currently supported platforms are Affymetrix (100k Hind and Xba, 250k NSP and STY, and SNP6.0) and Illumina (Infinium). For each Affymetrix chip type, probe maps from the corresponding BioConductor packages are used to match the probes for signal aggregation. A call to the function converts all CEL files of the same type (Affymetrix 250k NSP) in the current working folder. It is used as

```
> SCALA.convert(datatype=[type], calibrate=[T/F]),
                readfolder=getwd(), savefolder=getwd() )
```

Possible data types that are currently implemented are `Affy50kHind`, `Affy50kXba`, `Affy250kNSP`, `Affy250kSTY` and `AffySNP6.0`, which are self-explanatory. For successful conversion, if `SCALA.convert` is located in folder `X:/`, then the conversion maps should be placed in `X:/Maps`. Note that these files need to be of the same chip type; here all files are Affymetrix 250k NSP chips. Including other chips will provide errors.

For Illumina arrays we have a function that takes the X-raw and Y-raw columns. To use these arrays, the X and Y components in the SCALA object described below should be replaced with the X-raw and Y-raw columns from an Illumina data file. The chromosome allocation and position can be replaced similarly.

Genotype calls are all set by default to `NA` after file conversion but they can be added from any other source like HapMap, CRLMM or BirdSeed. Genotypes from HapMap have to be matched by SNP ids. This is because not all SNP ids in a sample are genotyped in HapMap with an identical id. Therefore, adding HapMap genotypes is also not (yet) automated. The required

format is a vector having AA=1, AB=2 and BB=3 for the genotype for each SNP, with SNPs and genotypes in the same order.

Precomputed calibration sets are also provided for a number of platforms, so that calibration can already be performed at the file conversion stage. However, it is also possible to convert a custom set of arrays, add genotypes, and then compute the calibration parameters from this new set with the function in section 7.2 and correct the signals manually.

Estimating SNP genotypes: SCALA.genotype

After file conversion genotype information is not available (NA), but can be obtained with the provided calling function. It requires a number of parameters. First, the number of bins (in both horizontal (`xbin`) and vertical (`ybin`) direction) for the histograms smoother has to be chosen. Second, the smoothing parameter λ has to be set, to determine the amount of smoothing in the histogram. Third, the initial vertical split levels for the three mixture components have to be set. These levels are defined in terms of the ratio of the number of vertical bins. If in Figure 7.3 the number of vertical bins is set to 100, the split levels for the left panel could be [0.45, 0.55] leading to splits at bin 45 and 55. Similarly, for the right panel initial split levels might be [0.50, 0.70]. The number of iterations is limited by `nit` and the convergence criterion is set by `crit`. After invoking the function, the `NA`s in `calls` are replaced with real genotypes by

```
> SCALA = SCALA.call(data=[name-of-CELfile], model=[],
                     plot=[T/F], save=[T/F], xbins=[val],
                     ybins=[val], lambda=[val], split1=[val],
                     savefolder=[])
```

As stated before, genotypes can also be added from another source (e.g. HapMap), but currently no function is provided to do so. If done manually, make sure that the ordering of the external genotypes matches the SNP ordering in the object. The column `SCALA$rsid` can be used for this purpose.

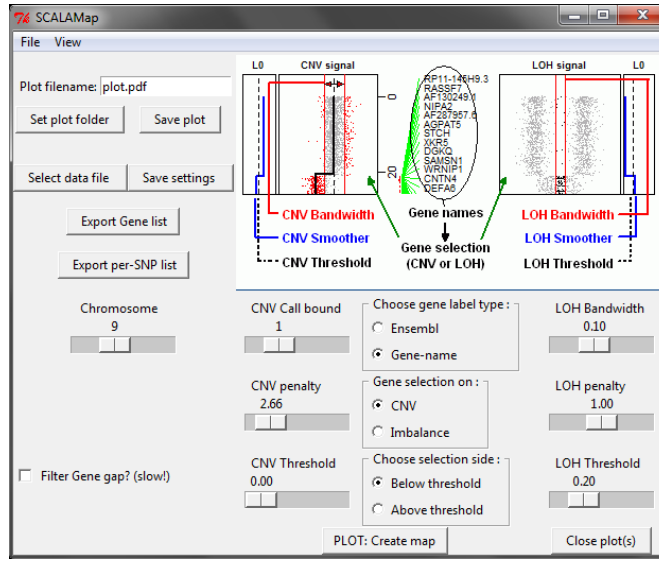

Figure 7.1: Graphical User Interface for SCALA.map.

Copy number mapping: SCALA.map

The mapping function set provides a novel way to combine visualization as well as analysis of both CNV and allelic imbalanceat the same time. The program is controlled solely by a graphical user interface (Figure 7.1) based on RPanel (Bowman et al, 2007). The GUI provides easy access to like chromosome selection, threshold signal filtering, the amount of L_0 smoothing, the type of gene labeling (e.g. taken from Ensembl or BiomaRt) to be used and some L_0 detection bandwidths and thresholds. These options can be saved to a(n `.Rdata`) file. It also provides the option to save the "tuned" image to PDF with a filename chosen by the user. Furthermore, it exports signals and detection results for all SNPs and genes located on the detected chromosome to a *comma-separated* table.

Input and output format

The mapping program takes a SCALA object as input. The main components of the object are shown below:

```
 $ meta :List of 6
  ..$ fname      : chr "name-of-CELfile"
  ..$ readpath   : chr "path-to-readfiles"
  ..$ savepath   : chr "path-to-savefiles"
  ..$ convertDate: chr "yyyy-mm-dd hh:mm:ss"
  ..$ calibrated : logi FALSE
  ..$ callDate   : logi NA
$ chr  : chr [1:numberofSNPs] "1" "1" "2" "3" ...
$ pos  : int [1:numberofSNPs] 101 102 103 104 ...
$ rsid : chr [1:numberofSNPs] "rsid1" "rsid2" "rsid3" ...
$ X    : int [1:numberofSNPs] val1 val2 val3 val4 ...
$ Y    : int [1:numberofSNPs] val1 val2 val3 val4 ...
$ calls: logi [1:numberofSNPs] NA NA NA NA NA NA NA ...
 - attr(*, "class")= chr "SCALA"
```

Settings are stored in an .controls object can be saved to an .Rdata file. These settings can be loaded upon GUI startup to recreate exactly the same plot as before, by using

```
> SCALA.map(controls = [settings_file])
```

The exported results contain gene-name, chromosome, start and stop location, a detection indicator (indicated by 0 or 1) that shows whether or not the gene was detected (by either the CNV or imbalance smoother), the accompanying L_0 values for the CNV and imbalance signal for each SNP as well as the smoothed CNV and imbalance values for each gene. Filenames for exported results as well as for the created PDF plot are user-controlled. A resulting plot window is shown in the example in section 7.3.

Graphical representation

At GUI startup, the plot window is not created immediately; the GUI starts with either the default settings or previous settings as specified by the user. The `PLOT: Create map` button is used to create and update the plot window after changing settings. The title contains the name of the selected sample and current chromosome by default, but it is highly adjustable via a separate window that is called from the menu. An partial plot example is included in the GUI, in which the main controls and their effect are illustrated. From left to right, the first panel shows the L_0 values for the CNV signal based on the data shown in the adjacent panel. In the CNV signal panel, the full CNV signal is given along with the L_0 smoother results. In the middle panel, selected gene names are shown for the chromosomal region(s) that show(s) abnormalities. The names can be shown for either the CNV or imbalance signal. To the right of the column with gene names, the imbalance signal is given (with an L_0 detection band), followed by the accompanying smoother based on the selected data points. Regions of aberrations are detected relative to a threshold value (dotted line) that is set by the user, as well as the level of smoothing and penalty norm power (default: $p = 0$) in the L_0 computations.

Estimating calibration parameters: SCALA.calibrate

The sets of calibration parameters based on a chosen number of arrays (located in the current working folder) are obtained (after file conversion) using

```
> params = SCALA.global(filefolder=getwd(), savefolder=getwd(),
                        filename=[nameofsavefile], kappa = 1e-8))
```

with `kappa` a small additive term to avoid singularity. The α (and β) vectors can be used to calibrate the original fluorescence signals.

7.3 Illustrative examples

In this example we use 8 high quality reference arrays from Affymetrix and one brain tumor file from the Erasmus Medical Center, Rotterdam, The

Netherlands (Bralten et al., 2010). The first are used to obtain calibration parameters, to be applied to the second. We start by setting the R working directory: `> setwd("D:/SCALASuite")`.

Obtaining calibration parameters

Place the Affymetrix 250k NSP control files in the folder `"D:/SCALASuite/01 raw/"` and create the folder `"D:/SCALASuite/02 raw/"`. Next we convert all CEL files in the first folder.

File conversion

We specify the chip type, read and save folder, as well as that *no* calibration should be applied; at this stage, calibration parameters are not yet available.

```
> source("SCALA.convert.r")
> SCALA.convert(datatype = 'Affy250kNSP', calibrate = F,
                readfolder = paste(getwd(),'/01 raw',sep = ""),
                savefolder = paste(getwd(),'/02 arrays',sep = ""))
Converting ctr aff 1.CEL
⋮
Converting ctr aff 8.CEL
```

Now that these files are converted and in stored R format, the next step is to call the genotypes for each array. After genotyping, it is possible to estimate the calibration parameters.

Genotype calling

A list of all converted files is obtained from the save directory defined above. Next the genotyping function is invoked. The semi-parametric estimation procedure is used, mixture plots for each chip are not requested, and the histogram is built from 100 by 100 bins.

```
> source("SCALA.genotype.r")
> fnames = list.files(path=paste(getwd(),"02 arrays",sep=""),
                      full.names=T)
> for (i in 1:length(fnames)) {
      load(fnames[i])
      scala = SCALA.genotype(scala=scala, model="s", plot=F, save=T,
                             xbins = 100, ybins = 100, lambda = 10,
                             nit=50, crit=1e-4, savefolder =
                             paste(getwd(),"/02 arrays",sep=""))
}
Calling ctr aff 1.CEL
⋮
Calling ctr aff 8.CEL
```

The (partial) result for the first of the 8 Affymetrix arrays shows that indeed the genotypes (and their cluster probabilities) have been added to the object and file, as well as the genotyping date.

```
> str(scala)
List of 8
 $ meta :List of 7
  ..$ fname      : chr "ctr aff 1.CEL"
  ..
  ..$ callDate   : chr "2011-09-12 22:16:46"
..
 $ calls: num [1:262264] 3 2 3 1 2 2 3 3 3 1 ...
 $ W    : num [1:262264, 1:3] 1.96e-10 2.15e-04 2.57e-08 1.00 ...
 - attr(*, "class")= chr "SCALA"
```

Calibration function

Now that genotypes are available, we estimate the calibration parameters for this chip type by

```
> source("SCALA.calibrate.r")
```

```
> params = SCALA.calibrate(
                              filefolder=paste(getwd(),"/02 arrays",sep=""),
                              savefolder=getwd(),
                              filename="scala.global.Rdata",
                              kappa=1e-8)
```

The generic result (object) has the following structure:

```
> str(params)

List of 7
 $ celfiles: chr [1:numberofFiles] "name-of-CELfile" ...
 $ alphaX  : num [1:numberofSNPs] num1 num2 num3 ...
 $ alphaY  : num [1:numberofSNPs] num1 num2 num3 ...
 $ betaX   : num [1:numberofFiles] num1 num2 num3 ...
 $ betaY   : num [1:numberofFiles] num1 num2 num3 ...
 $ gammaX  : num [1:3] num1 num2 num3
 $ gammaY  : num [1:3] num1 num2 num3
```

We then extract and store the calibration parameters for later use:

```
> alphaX = params$alphaX; alphaY = params$alphaY
> save(alphaX, alphaY, file = paste(getwd(),"/Calibration/",
                                    "Affy250kNSP.Rdata",sep="")
```

Copy numbers in a new glioblastoma array

The vectors just saved will be applied to a tumor tissue chip.

```
> SCALA.convert(datatype = 'Affy250kNSP', calibrate = T,
                readfolder = paste(getwd(),'/01 raw',sep = ""),
                savefolder = paste(getwd(),'/02 arrays',sep = ""))
Converting GBM 139.CEL
```

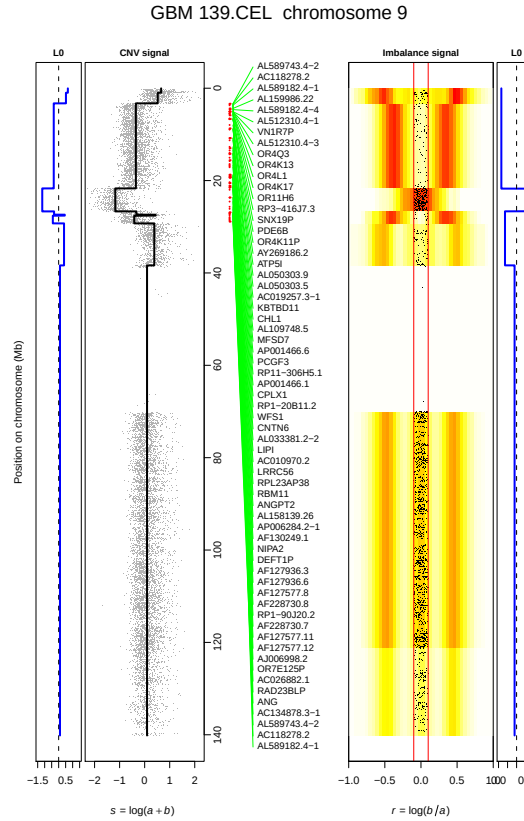

Figure 7.2: Example of a SCALA.map image. On the far left, it shows the CNV L_0 signal smoother and to its right the raw CNV signal. The middle part shows the selected gene names. The rightmost parts show the signal for allelic imbalance and its L_0 signal smoother. Here, SCALA.Map is set here to detect aberrated regions on the CNV signal, identifying one problematic region and the genes it contains.

After file conversion, we start the GUI with predefined settings (in `SCALA.map-controlsGBM139.Rdata'`). To obtain the plot window (and implicitly perform all computations), click the appropriate button.

```
> source("SCALA.map.r")
```

```
> SCALA.map(controls='SCALA.map-controlsGBM139.Rdata')
Selections:  ..  done!
Plotting:  ..  done!
Computations:  ..  done!
```

The resulting plot for chromosome 9 is given in Figure 7.2. It shows a small CNV region that has less than 2 alleles present. 27 genes are contained in that region of chromosome 9. The white space is the centromere. A similar detection can be performed on the imbalance signal by simply changing the GUI options to this purpose. Saving the settings used to obtain the current plot can be done by clicking the "save settings" button. Exporting the detection show in the plot can be either per gene or per SNP, depending on the selected button. For button location, see Figure 7.1.

7.4 Technical model details

Semi-parametric genotyping

Let $Y = \{y_{ih}\}$ be an $n_1 \times n_2$ matrix of counts in a two-dimensional $n_1 \times n_2$ histogram. The center of bin (i, h) is given by (u_i, v_h). The expected values are modeled by sums of tensor product B-splines. Two bases are computed, B_1, with c_1 columns, based on u and B_2, with c_2 columns, based on v. The bases are combined with a $c_1 \times c_2$ matrix Θ of coefficients, and the matrix of expected values is computed as

$$M = \exp(B_1 \Theta B_2'). \tag{7.1}$$

A penalized Poisson log-likelihood is then optimized. The penalty is complex, because both rows and columns of Θ are penalized. If $||X||_F$ indicates the Frobenius norm of the matrix X, i.e. the sum of the squares of its elements, the penalty is

$$\text{Pen} = \lambda_1 ||D_1 \Theta||_F / 2 + \lambda_2 ||\Theta D_2'||_F / 2, \tag{7.2}$$

where D_1 and D_2 are matrices of the proper dimensions ($c_1 - 3 \times c_1$ and $c_2 - 3 \times c_2$) that form third differences.

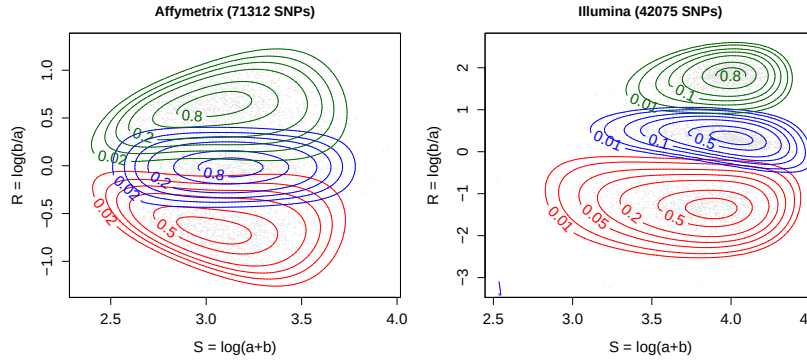

Figure 7.3: Raw data with estimated smooth densities. Left panel: a typical (symmetric) Affymetrix array. Right: a typical (asymmetric) Illumina array. Contours, normalized to 1, are overlayed for [0.02,0.05,0.1,0.2,0.5,0.8] (left) and [0.01,0.02,0.05,0.1,0.2,0.5,0.8] (right).

The mixture components give three expected values for bin (i, h) of the histogram: μ_{ih1}, μ_{ih2} and μ_{ih3}. From these numbers follow, after division by their sum, three membership probabilities. The largest of the three, which we indicate by $\hat{p}_{ih}$ points to which cluster all the observations in the bin should be assigned. The result of this algorithm is depicted in Figure 7.3, where contours of the fitted densities are shown for an Affymetrix (left) and Illumina (right) array.

In Rippe et al. (2010) genotype calls from a multi-array method (CRLMM) and from our single-array method (SCALA) are compared to a set of consensus genotypes from HapMap. The number of agreements and differences in terms of homo- and heterozygous calls showed that SCALA can be used to call genotypes efficiently and effectively. Even SNPs that were not genotyped in HapMap can be genotyped with reasonable certainty using a single chip. The above model is implemented in the SCALA.call function.

Copy number signal smoother

To obtain smooth estimates of the data, a method derived from Eilers and DeMenezes (2005) is applied. In Rippe et al. (2012b), the algorithm of Eilers and DeMenezes has been improved in at least two ways: 1) a least squares measure of fit to increase sensitivities and 2) and an L_0 norm penalty to reduce the number of jumps. The formal model is given in (7.3).

$$S_q = \sum_{i=1}^{m} (y_i - z_i)^2 + \lambda \sum_{i=2}^{m} |z_i - z_{i-1}|^q \tag{7.3}$$

with y the original data and z the smoothed signal. λ is again a tuning parameter that controls amount of smoothness. Furthermore, q can be any number between 0 and 2. Here, $q = 0$, essentially a penalty on the number of non-zero difference between neighboring elements of z, while in Eilers & DeMenezes (2005), $q = 1$.

For CNV signal smoothing all observations are used. For allelic imbalance a selection of observations within a user-defined bandwidth is used.

SNP signal intensity calibration

Consider one allele. Assuming that SNP i has a specific intensity level a_i, and that array j has a normalization factor b_j, a reasonable model for the intensity fluorescence signal is given by $x_{ij} = a_i b_j u_{ij} + \epsilon_{ij}$, with $i = 1, ..., m$ and $j = 1, ..., n$ and where u_{ij} represents the number of copies of the allele (0, 1 or 2) and ϵ_{ij} represents the error. We do not specify a distribution of ϵ. Instead we will use signals on a logarithmic scale (base 10), with $y_{ij} = \log x_{ij}$. A similar model holds for the other allele.

For the moment we assume the genotypes to be given, so we can formulate a linear model:

$$y_{ij} = \mu + \alpha_i + \beta_j + \sum_{k=1}^{3} \gamma_k h_{ijk} + e_{ij}. \tag{7.4}$$

where μ is the grand mean, α_i the level of SNP i, and β_j the level of array j; k indexes the genotype ($1 = AA$, $2 = AB$, $3 = BB$) and γ_k is a parameter for

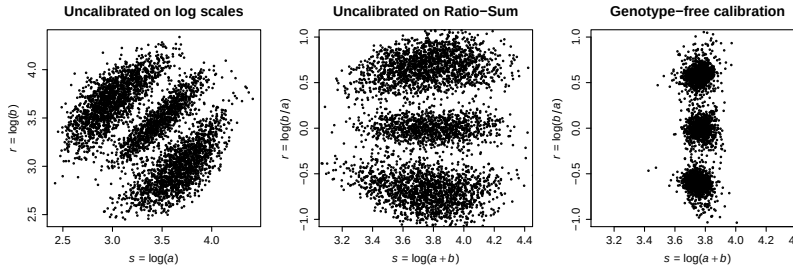

Figure 7.4: Calibration results for chromosome 1 in array NA06985 from the HapMap CEU population. Left: signals before calibration. Right: signals after genotype-free calibration. In the right panel, note the change in x-axis range, after calibration.

genotype k. The genotypes are coded with the indicator array $H = \{h_{ijk}\}$; for each combination of i and j we have a 1 in the array cell that corresponds to the genotype k, and 0 in the other cells. To make the model identifiable we introduce the constraints $\sum_i \alpha_i = 0$ and $\sum_j \beta_j = 0$. We call this the global model, since all SNPs share the same genotype parameters γ.

Further theoretical and technical details on the implementation of model (7.4) are discussed in Rippe et al., (2012a).

The α parameters that represent the levels of the SNPs are used to calibrate the intensities on a new array, by computing $x^*_{ij} = x_{ij}/10^{\alpha}_i$. This is done for each allele separately. Note that genotypes are not used, hence we call it 'genotype-free calibration'.

Figure 7.4 illustrates the results of calibration for chromosome 1 in an Affymetrix 100k Hind sample (NA06985) from the complete CEU population, retrieved from the HapMap database (HapMap Consortium, 2007). We show the signal combination $\log(a + b)$ against $\log(b/a)$, for all SNPs in a single chip. It is clear that a strong reduction of the 'noise' is obtained.

7.5 Discussion

We have described a set of programs that perform signal calibration, genotyping and copy number mapping for individual SNP chips. In situations with novel species and chips (Wright et al., 2010) this approach can be very useful: the first available chip can be genotyped or inspected for copy numbers immediately.

When genotyping, the software uses a sum-and-ratio transformation ($x = \log(a+b)$ and $y = \log(b/a)$) of the raw signals, before computing the smoothed 2D histogram. These transformations are hard coded. It can however be argued that other transformations can or should be used. With respect to the horizontal axis, $x = \log(a \times b)$ can be used, for example. We state that this alternative doesn't influence the genotype calls, since it only stretches the observations in horizontal direction. Furthermore, this algorithm calls genotypes for a whole chip (all chromosomes) at once. A discussion of genotyping individual chromosomes versus the whole genome is presented in Rippe, Eilers & Meulman (2010).

SCALA.Map provides a method to map SNPs on their chromosomal position both visually and statistically, using signals that indicate CNV and allelic imbalance. It integrates visual analysis with L_0 signal smoothers, which are all user-controlled and very easy to use. Quantification of the CNV or allelic imbalance regions is implicitly done via the L_0 norm. Furthermore, the software has several customization and export options. Intended additions and extensions are probability-based signal thresholding (i.e. to remove 'rubbish' signal for low quality samples by taking only signals above a user-defined threshold value into account). Cross-validation to determine the optimal amount of smoothing as well as a segmented scatterplot (both described in Rippe et al., 2012b)). are candidates for implementation. Furthermore, in the same paper a method for segmented imbalance estimation using segment-wise mixtures is proposed. We feel that this idea still deserves further attention before final implementation.