

```
### --- BIOINFORMATICS TRAINING SCHOOL (EuGESMA) --- ###
### --- 23-24 Marzo 2010 - Barcelona - Spain --- ###
```

```
# 3rd SESSION:
# Microarray data analysis:
SNP arrays applied to calculate Copy Number alterations in cancer samples.
# Celia FONTANILLO and Dr. Javier DE LAS RIVAS (cfontanillo@usal.es; jrivas@usal.es)
# Centro de Investigacion del Cancer, CiC-IBMCC, CSIC/USAL, Salamanc, Spain
```

LAB: Analysis of Copy Number and Genotyping using human SNP arrays (aroma.affymetrix on 500K tumor data) (CRMAv2, CBS)

This lab session will teach you how to perform low-level CN analysis on a set of tumor samples.

Step 1: Setup

Required file structure

A standard directory structure of an aroma project looks like this:

```
<current working directory>
+- annotationData/
| +- chipTypes/
|   +- <chipTypeA>/ <-- must match exactly the name of the CDF file
|   | +- CDF file(s) and other annotation (possibly subdirectories)
|   |
|   +- <chipTypeB>/ <-- must match exactly the name of the CDF file
|   | +- CDF file(s) and other annotation (possibly subdirectories)
|   |
|   ...
|
+- rawData/
| +- <dataSet1>/
| | +- <chipTypeA>/ <-- must match exactly a chip type folder under annotationData/
| | | +- CEL files
| |
| +- <dataSet2>/
| | +- <chipTypeB>/ <-- must match exactly a chip type folder under annotationData/
| | | +- CEL files
| |
| +- <dataSet3>/
| | +- <chipTypeA>/ <-- must match exactly a chip type folder under annotationData/
| | | +- CEL files
| |
| ...
```

You must follow this structure/format exactly including the case of letters. For instance, 'GenomeWideSNP_6' is not the same as 'genomewidesnp_6'. The <...> marks indicate a name of folder or file that you must supply according to your project/computer/chip type etc.

Annotation data

- **chip definition file (CDF)**: A CDF contains information on which probes belong to what probeset, the (x,y) location of each probe, which the middle nucleotides in the target and the probe are (from which PM/MM status is inferred). This file can be downloaded from:
<http://www.affymetrix.com/support/technical/libraryfilesmain.affx?highlight=true&rootCategoryId=>

- **unit fragment length file (UFL):** Aroma UFL files are binary files storing a unit-fragment length map in a tabular format. The number of fragment lengths may be more than one depending on how many enzymes were used in the assay. Can be downloaded from: <http://groups.google.com/group/aroma-affymetrix/web/mapping250k-nsp-mapping250k-sty>
- **unit genome position file (UGP):** Aroma UGP files are binary files storing a unit-chromosome-position map in a tabular format. Can be downloaded from: <http://groups.google.com/group/aroma-affymetrix/web/mapping250k-nsp-mapping250k-sty>
- **aroma cell sequence file (ACS):** Aroma ACS files are binary files storing probe sequences for a particular chip type. It contains a cell-probe-sequence-strand map in a tabular format. Can be downloaded from: <http://groups.google.com/group/aroma-affymetrix/web/mapping250k-nsp-mapping250k-sty>

In our example the annotation data folder will be as follow:

```
annotationData/
  chipTypes/
    Mapping250K_Sty/
      Mapping250K_Sty.cdf
      Mapping250K_Sty,na26,HB20080916.ufl
      Mapping250K_Sty,na26,HB20080915.ugp
      Mapping250K_Sty,HB20080710.acs
    Mapping250K_Nsp/
      Mapping250K_Nsp.cdf
      Mapping250K_Nsp,na26,HB20080916.ufl
      Mapping250K_Sty,na26,HB20080915.ugp
      Mapping250K_Sty,HB20080710.acs
```

Raw data

```
rawData/
  test/
    Mapping250K_Sty/
      (4 sample CEL files)
    Mapping250K_Nsp/
      (4 sample CEL files)
```

There should be 8 CEL files in total. These files are obtained from the Affymetrix sample data set for copy number analysis consisting of 9 Tumor/Normal pairs derived from human cancer cell lines (http://www.affymetrix.com/support/technical/sample_data/copy_number_data.affx). We have chosen 4 samples (CRL-2337D, CRL-2336D, CRL-2339D and CRL-2338D) to make all computations faster.

Step 2: Startup

```
library("aroma.affymetrix")
log <- verbose <- Arguments$getVerbose(-8, timestamp=TRUE)
chipTypes <- c("Mapping250K_Nsp", "Mapping250K_Sty")
```

Verifying annotation data files

Before we continue, the following asserts that the annotation files (CDF, UGP, UFL, and ACS) can be found. This test is not required, because aroma.affymetrix will located them in the background, but it will help troubleshooting if there is any problem.

```
cdf <- gi <- si <- acs <- list()
```

```

for (chipType in chipTypes) {
  cdf[[chipType]] <- AffymetrixCdfFile$byChipType(chipType) # .CDF
  gi[[chipType]] <- getGenomeInformation(cdf[[chipType]]) # .UGP
  si[[chipType]] <- getSnpInformation(cdf[[chipType]]) # .UFL
  acs[[chipType]] <- AromaCellSequenceFile$byChipType(getChipType(cdf[[chipType]]),
fullname=FALSE)) # .ACS
}

```

Q: How many probes are there on this chip type?

Q: How many units (loci) are there?

```

print(cdf[["Mapping250K_Nsp"]])
print(gi[["Mapping250K_Nsp"]])
print(si[["Mapping250K_Nsp"]])
print(acs[["Mapping250K_Nsp"]])

```

Step 3: Setup raw data set

First thing to do for a paired analysis is to **define the samples** and which the reference is going to be.

```

dataSetName <- "test"
pairs <- matrix(c(
  "CRL-2337D", "CRL-2336D",
  "CRL-2339D", "CRL-2338D"
), ncol=2, byrow=TRUE)
colnames(pairs) <- c("Normal", "Tumor")

```

```

      Normal      Tumor
[1,] "CRL-2337D" "CRL-2336D"
[2,] "CRL-2339D" "CRL-2338D"

```

As we have two different chips (one for each enzyme used, **Nsp & Sty**) we need to read both chips.

Moreover, the names have to be changed adding the enzyme type as a tag:

from CRL-2337D_NSP.CEL to CRL-2337D,Nsp.CEL

from CRL-2337D_STY.CEL to CRL-2337D,Sty.CEL

Defining CEL set:

```

csRawList <- list()
for (chipType in chipTypes) {
  cs <- AffymetrixCelSet$byName(dataSetName, cdf=cdf[[chipType]])
  setFullNamesTranslator(cs, function(names, ...) {
    names <- gsub("_", "", names)
    names <- gsub(",(Nsp1|Sty1)", "", names)
    names
  })
  stopifnot(all(getNames(cs) %in% pairs))
  csRawList[[chipType]] <- cs
}
print(csRawList[["Mapping250K_Nsp"]])

```

AffymetrixCelSet:

Name: test

Tags:

Path: rawData/test/Mapping250K_Nsp

Platform: Affymetrix

Chip type: Mapping250K_Nsp

Number of arrays: 4

Names: CRL-2336D, CRL-2337D, CRL-2338D, CRL-2339D

Time period: 2006-01-13 16:19:29 -- 2006-01-13 20:52:36

- Q: What is the name of the first Sty array?
 Q: During what period were the arrays hybridized/scanned?

Initial quality assessment

The very first thing one might want to check with a new data set is to see if there are arrays that are obviously bad or behaving rather different from the others. Here we will look at the empirical distribution of probe intensities as well as spatial plots of the probe intensities.

Density plots

In order to a better compation of the two array types, we will plot both on the same window:

```
# Plot the PM densities for the raw data set:
jpeg("RawDataDensities.jpg", width=600, height=400)
par(mfrow=c(1,2)) # Create two vertical panels on the same window
for (chipType in chipTypes) {
  plotDensity(csRawList[[chipType]], types="pm", ylim=c(0,0.4))
}
dev.off()
```

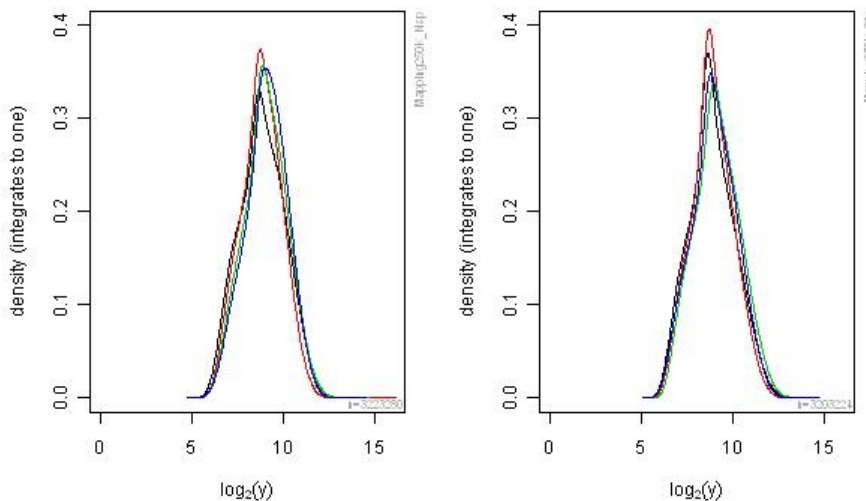


Figure 1: RawDataDensities.jpg

- Q: Can you see the density curve for each array of the two chip types?
 Q: Is there any array that is much different from the others?

Spatial plots (DO NOT RUN, TAKES TOO LONG !!!)

We can generate spatial images of the probe intensities for all arrays and the browse them in Firefox using the **ArrayExplorer**. As this function may take too long, the corresponding output files have been located in the material web site (in the reports/test/raw folder).

```
# Setup an ArrayExplorer for the raw data set and use a black-to-yellow color ramp to display square-root
# transforms probe intensities (this transform turns out to useful for the eye):
for ( chipType in chipTypes) {
  ae <- ArrayExplorer(csRawList[[chipType]])
  setColorMaps(ae, "sqrt,yellow")
  # Now, generated the image files
  process(ae, verbose=log)
  # When done, have R open the ArrayExplorer in the browser
```

```
  display(ae)
}
```

Q: Can you spot any spatial artifacts?

Step 4: Calibration for crosstalk between allele probe pairs (CRMA v2, 1st step)

The first thing we will do is calibrate the probe signals such that **crosstalk** between allele A and allele B of each SNP is removed. **Offset** is also removed from the CN probes.

Allelic crosstalk calibration is a single-array method, that is, each array is calibrated independently of the others. This means that you can use this method to calibrate a single array and having more arrays will not make difference.

For this step you will need You will need the CDF file and the probe sequences in the .ACS file.

The default model is to use CRMA v1, which is why we have to explicitly specify that we want to use the crosstalk model of **CRMA v2**.

```
csR <- csRawList[["Mapping250K_Sty"]]
acc <- AllelicCrosstalkCalibration(csR, model="CRMAv2")
print(acc)
```

Q: What name and the tags will the calibrated data set get?

Q: In what subdirectory will it be stored?

It is important to understand that the above code *does not* calibrate the data set, but instead it only sets up the method for doing so. We will do that in the following step.

```
csC <- process(acc, verbose=log)
print(csC)
```

As we have two chip types it is necessary to process not only Nsp data but also Sty data. The following chunk of code will process both chip types.

```
csAccList <- list()
for (chipType in names(csRawList)){
  csR <- csRawList[[chipType]]
  acc <- AllelicCrosstalkCalibration(csR, model="CRMAv2")
  csAcc <- process(acc, verbose=log)
  csAccList[[chipType]] <- csAcc
}
```

Next, we will check if the calibration made any difference by plotting the six different (A/C, A/T, C/T, A/G, C/G, G/T) allele-pair probe signals (PMA,PMB). We will plot the probe signal densities before (we will use "input" as parameter) and after ("output" parameter) calibration for offset and crosstalk to observe the difference

```
# Plot non-calibrated (PMA,PMB):
xlim <- c(-500,15000)
jpeg("non-calibrated.jpg", width=800, height=580)
plotAllelePairs(acc, array=4, pairs=1:6, what="input", xlim=xlim/3)
dev.off()
```

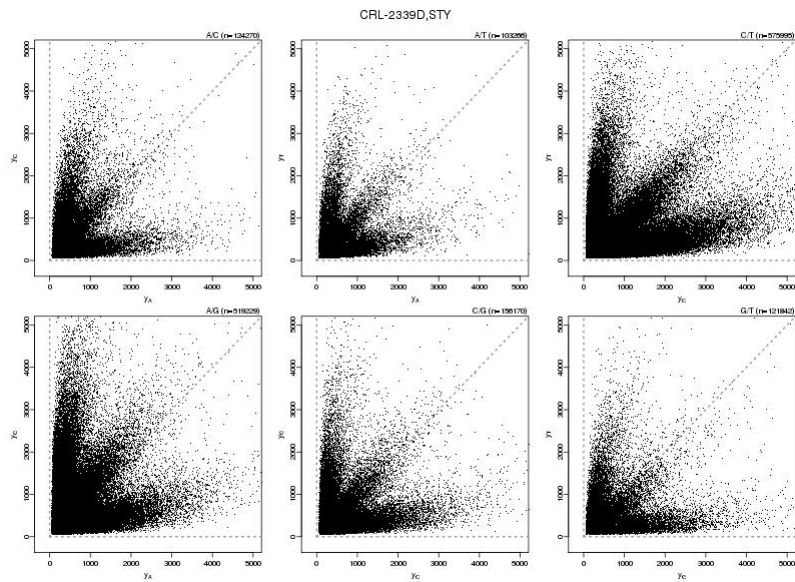


Figure 2: non-calibrated.jpg

```
# Plot calibrated (PMA,PMB):
jpeg("calibrated.jpg", width=800, height=580)
plotAllelePairs(acc, array=4, pairs=1:6, what="output", xlim=xlim)
dev.off()
```

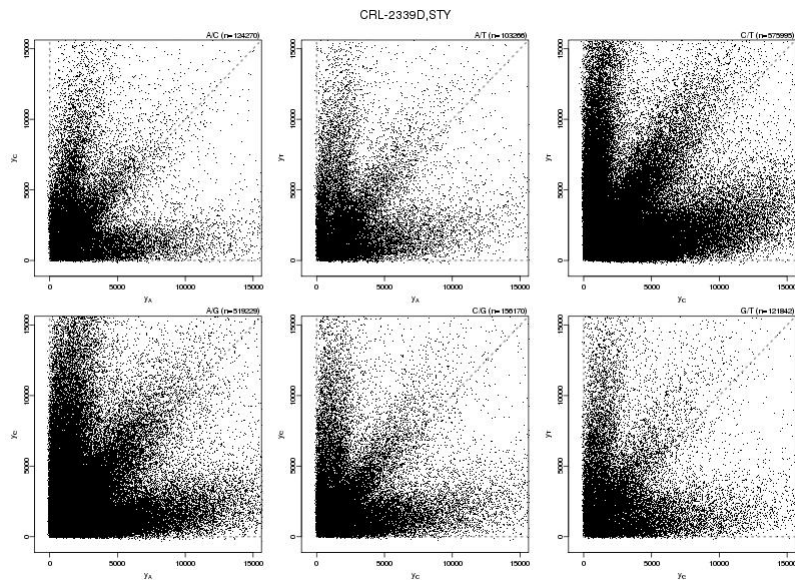


Figure 3: calibrated.jpg

Q: Are the crosstalk calibrated signals more "orthogonal" than the raw signals?

Density plots

Go back to **Figure 1** and plot the densities for the calibrated probe signals:

```
# Plot the PM densities for the calibrated data set:
jpeg("ACCDensities.jpg", width=600, height=400)
par(mfrow=c(1,2)) # Create two vertical panels on the same window
for (chipType in chipTypes) {
  plotDensity(csAccList[[chipType]], types="pm", ylim=c(0,0.4))
}
dev.off()
```

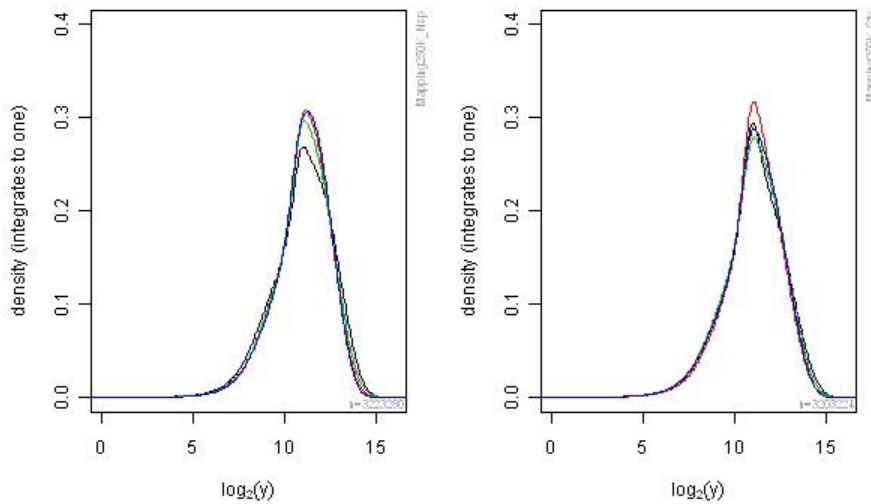


Figure 4: ACCDensities.jpg

Q: What do you observe?

Step 5: Normalization for nucleotide-position probe sequence effects (CRMA v2, 2nd step)

After calibrating for allelic crosstalk, we will control for variability due to **probe sequence effects**. The affinity of a probe can be attributed to its sequence composition. Taking this into account, in this step, we model the probe sequence affinity as a function of nucleotide and position. Probe sequence effects differ slightly across arrays and removing them will control for variability across arrays. Importantly, the effects also differ between PMA and PMB, so this normalization will also correct for imbalances between alleles for heterozygote data points.

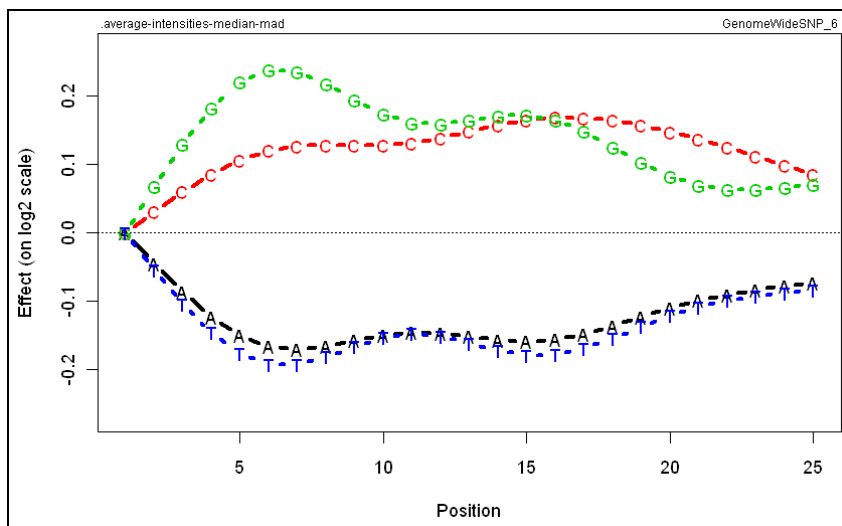


Figure 5: Nucleotide position model for an

For this step you will also need the CDF file and the probe sequences in the .ACS file.

Start by setting up the nucleotide (base) position normalization method such that it normalizes the crosstalk calibrated signals. If target has "zero" value, all arrays are normalized to have no effects and if it has "NULL" value (default value), all arrays are normalized to have the same effect as the average array has.

```
csBPNList <- list()
for (chipType in names(csAccList)) {
  csC <- csAccList[[chipType]]
  bpn <- BasePositionNormalization(csC, target="zero")
  csBPN <- process(bpn, verbose=log)
  csBPNList[[chipType]] <- csBPN
}
```

When done, lets look at the probe densities of this crosstalk calibrated and sequence normalized data set:

```
# Plot the PM densities for the calibrated data set:
jpeg("BPNDensities.jpg", width=600, height=400)
par(mfrow=c(1,2)) # Create two vertical panels on the same window
for (chipType in chipTypes) {
  plotDensity(csBPNList[[chipType]], types="pm", ylim=c(0,0.4))
}
dev.off()
```

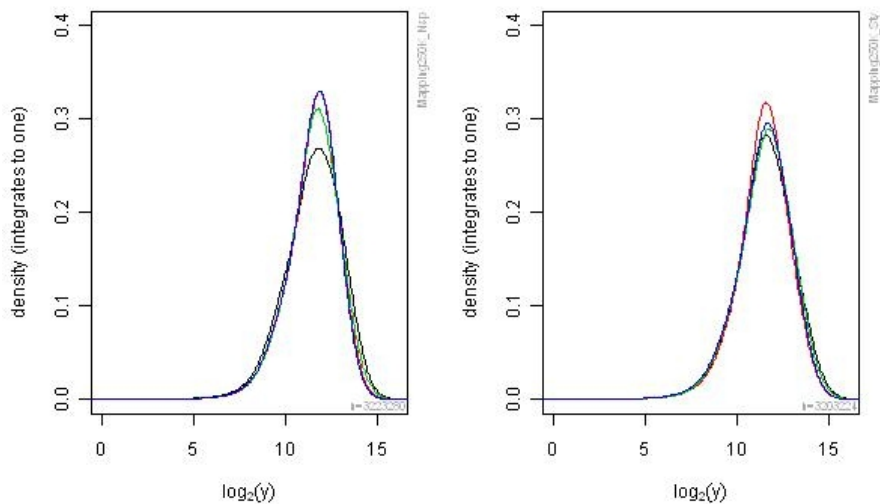


Figure 6: BPNDensities.jpg

Step 6: Probe summarization (CRMA v2, 3rd step)

Next, we summarize the probe level data unit by unit. For SNPs we have the option to model either the total CN signals (**combineAlleles**=TRUE) or allele-specific signals (**combineAlleles**=FALSE). Here we choose to fit allele-specific CN estimates, because we can always get total CN signals downstream.

All 10K, 100K and 500K chip types have SNP probes that interrogate both strands, direct and reverse. We are not interested in differentiation between these (**mergeStrands**=TRUE).

Finally, since the replicated SNP probes on these arrays has slightly shifted sequences (-4 to +4 shift in nucleotides), they have different probe sequences and therefore different binding properties. For this reason, we will here use a probe summarization method that takes that into account, **RmaCnPlm**. This is known as the log-additive model that controls for different probe affinities. For the GenomeWideSNP_5 and GenomeWideSNP_6 chip types all replicated probes in a probe set are identical technical replicates which

we expect to have the same probe affinities. That is why it is preferable to take the robust average without modeling probe affinities: **AvgCnPlm** (for plain averaging).

As far as this step takes too long and `aroma.affymetrix` package stores all results in persistent memory, i.e. on the file system we have taken this step preprocessed in `<working_directory>/plmData/test,ACC,-XY,BPN,-XY,RMA`. If we execute this chunk of code the package will detect the fit is already done and quickly return the existing results.

This function also generate CDF files with the new summarized information: `Mapping250K_Nsp,monocell.CDF` and `Mapping250K_Sty,monocell.CDF`. This files are also previously created and must be placed on the respective annotations folders.

```
plmList <- list()
for (chipType in names(csBPNList)) {
  cs <- csBPNList[[chipType]]
  plm <- RmaCnPlm(cs, mergeStrands=TRUE, combineAlleles=FALSE)
  fit(plm, verbose=log)
  plmList[[chipType]] <- plm
}
print(plmList[["Mapping250K_Sty"]])
```

Step 7: Normalization for PCR fragment-length effects (CRMA v2, 4th step)

In this step we will correct for the PCR fragment length effects. Longer fragments are amplified less by PCR, what makes the observed signals weaker

But before beginning this **normalization** step it is necessary to recover the **chipEffectSet** data. Before this point we have been using **AffymetrixCelSet** objects with probe intensities, but after summarization step we will be using **chipEffectSet** objects that contain the intensities for every SNP (thetaA and thetaB).

```
cesList <- list()
for (chipType in names(plmList)){
  cesList[[chipType]] <- getChipEffectSet(plmList[[chipType]])
}
print(cesList[["Mapping250K_Sty"]])
```

CnChipEffectSet:

```
Name: test
Tags: ACC,-XY,BPN,-XY,RMA
Path: plmData/test,ACC,-XY,BPN,-XY,RMA/Mapping250K_Sty
Platform: Affymetrix
Chip type: Mapping250K_Sty,monocell
Number of arrays: 4
Names: CRL-2336D, CRL-2337D, CRL-2338D, CRL-2339D
Time period: 2010-03-17 22:23:53 -- 2010-03-17 22:23:53
```

Similarly to how we normalized for the probe-sequence effects above, we will here normalize for PCR fragment-length effects by using a "zero" target. This will avoid using the average (chip effects) as a reference. Also we will need Affymetrix CDF together with the Unit Fragment Length file (UFL).

```
cesFlnList <- list()
for (chipType in names(cesList)) {
  ces <- cesList[[chipType]]
  fln <- FragmentLengthNormalization(ces, target="zero")
  cesFln <- process(fln, verbose=log)
  cesFlnList[[chipType]] <- cesFln
}
print(cesFlnList[["Mapping250K_Sty"]])
```

Lets do the same to the total SNP signals, i.e. **combine alleles** $\theta = \theta_A + \theta_B$. In order to do this, we first have to summarize the signals (AlleleSummation) as follows:

```
cesFlnTotalList <- list()
for (chipType in names(cesFlnList)){
  asN <- AlleleSummation(cesFlnList[[chipType]])
  cesFlnTotalList[[chipType]] <- process(asN, verbose=log) # This works only since
aroma.affymetrix_1.0 version
}
```

Step 8: Copy number segmentation (CBS)

In this section we will identify CN regions using the **Circular Binary Segmentation (CBS)** method. Current aroma.affymetrix CBS method is implemented for the total SNP signals, not for allele specific. We will use cesFlnTotalList with summarized signals instead the above cesFlnList object that contains (allele-specific) chip-effect estimates.

Before using this segmentation method it is necessary to put together in an object the total chip-effect estimates from the two chips we are using. This step is only required for 10K, 100K and 500K chip types with two enzymes hybridized separately. As we are carrying out a paired analysis (i.e. **reference** samples *versus* **studied** samples) it is also important to define our references. Here we will consider the normal blood samples as our references.

```
sets <- list(Normal=list(), Tumor=list())
for (chipType in names(cesFlnTotalList)) {
  ces <- cesFlnTotalList[[chipType]]
  for (type in colnames(pairs)) {
    idxs <- match(pairs[,type], getNames(ces))
    sets[[type]][[chipType]] <- extract(ces, idxs)
  }
}
```

```
$Normal
$Normal$Mapping250K_Nsp
CnChipEffectSet:
Chip type: Mapping250K_Nsp,monocell
Number of arrays: 2
Names: CRL-2337D, CRL-2339D
```

```
$Normal$Mapping250K_Sty
CnChipEffectSet:
Chip type: Mapping250K_Sty,monocell
Number of arrays: 2
Names: CRL-2337D, CRL-2339D
```

```
$Tumor
$Tumor$Mapping250K_Nsp
CnChipEffectSet:
Chip type: Mapping250K_Nsp,monocell
Number of arrays: 2
Names: CRL-2336D, CRL-2338D
```

```
$Tumor$Mapping250K_Sty
CnChipEffectSet:
Chip type: Mapping250K_Sty,monocell
Number of arrays: 2
Names: CRL-2336D, CRL-2338D
```

For CBS paired analysis the reference is given by the second parameter.

```
cbs <- CbsModel(sets$Tumor, sets$Normal)
fit(cbs, chromosomes=c(1:23), verbose=log)
print(cbs)
```

```

CbsModel:
Name: test
Tags: ACC,-XY,BPN,-XY,RMA,FLN,-XY,SA,paired
Chip type (virtual): Mapping250K_Nsp+Sty
Path: cbsData/test,ACC,-XY,BPN,-XY,RMA,FLN,-XY,SA,paired/Mapping250K_Nsp+Sty
Number of chip types: 2

```

To get a graphical image of the results we can use **ChromosomeExplorer**. As this function may take too long, the corresponding output files have been located in the material web site (in the reports/test/ACC,-XY,BPN,-XY,RMA,FLN,-XY,SA,paired folder).

```

ce <- ChromosomeExplorer(cbs)
process(ce, chromosomes=c(1,2), verbose=log)
display(ce)

```

Step 9: Calculating raw copy numbers & raw genotypes (for each SNP)

In this section we will show how to manually calculate raw copy numbers relative to a reference. Note that several of the downstream methods, such as segmentation method above, will do this automatically/internally and therefore it is often not necessary to do this manually.

Here we will look at Chromosome 5 in Array #1 (CRL-2336D). To get the chip effects for array #1, do:

```

array <- 1
chr <- 5
ce <- getFile(cesFlnList[["Mapping250K_Nsp"]], array)
print(ce)

```

```

CnChipEffectFile:
Name: CRL-2336D
Tags: NSP,chipEffects
Full name: CRL-2336D,NSP,chipEffects
Pathname: plmData/test,ACC,-XY,BPN,-XY,RMA,FLN,-XY/Mapping250K_Nsp/CRL-2336D,NSP,chipEffects.CEL
File size: 9.56 MB (10020706 bytes)
RAM: 0.02 MB
File format: v4 (binary; XDA)
Platform: Affymetrix
Chip type: Mapping250K_Nsp,monocell
Timestamp: 2010-03-09 18:41:21
Parameters: (probeModel: chr "pm", mergeStrands: logi TRUE, combineAlleles: logi FALSE)

```

Note that this is now referring to a single array, not the whole data set.

Next we identify which units (SNPs) are in that region and what their genomic positions are. We do this via the genome information (in the UGP file). We will do this just for one chip type (Mapping250K_Sty) but could be extended for both chip types.

```

gi <- getGenomeInformation(cdf[["Mapping250K_Sty"]])
units <- getUnitsOnChromosome(gi, chromosome=chr)
unitNames <- getUnitNames(cdf[["Mapping250K_Sty"]], units=units)
pos <- getPositions(gi, units=units) / 1e6

```

Q: How many loci are there in this chromosome? Hint, try `str(units)`.

Next we will transform our allele signals (θ_A y θ_B) into the total signal and the allele frequency estimates of this region for this array (θ , freq_B) = ($\theta_A + \theta_B$, θ_B / θ).

```

data <- extractTotalAndFreqB(ce, units=units)
str(data)

```

```
data[1:3,]
  total      freqB
[1,] 2517.258 0.27909055
[2,] 2641.608 0.64209158
[3,] 1501.706 0.85872248
```

Q: How many rows and columns are there in the 'data' matrix? What does the rows and the columns corresponds to?

Extract the (total) theta and the freqB columns:

```
theta <- data[,"total"]
freqB <- data[,"freqB"]
```

In order to calculate copy numbers, we need a reference. We will use for that the corresponding normal blood sample, CRL-2337D (array number #2).

```
ref_array <- 2
ceR <- getFile(cesFlnList[["Mapping250K_Nsp"]], ref_array)
dataR <- extractTotalAndFreqB(ceR, units=units)
thetaR <- dataR[,"total"]
```

Then the total copy numbers will be:

```
C <- 2*theta/thetaR
```

Finally, we plot the (C,freqB) along the genome:

```
par(mfrow=c(2,1))
xlim <- range(pos, na.rm=TRUE)
plot(pos, C, xlim=xlim, ylim=c(0,4), xlab="pos", ylab=expression(C), pch=".")
title(main=getName(ce))
stext(side=3, pos=1, paste("Chr",chr,sep=" "))
plot(pos, freqB, xlim=xlim, ylim=c(0,1), xlab="pos", ylab=expression(theta[B]/theta),
pch=".")
```

References:

H. Bengtsson, R. Irizarry, B. Carvalho & T.P. Speed, **Estimation and assessment of raw copy numbers at the single locus level.** *Bioinformatics* (2008) 24: 759-767.

H. Bengtsson, P. Wirapati & T.P. Speed, **A single-array preprocessing method for estimating full-resolution raw copy numbers from all *Affymetrix* genotyping arrays including GenomeWideSNP 5 & 6.** *Bioinformatics* (2009) 17: 2149-2156.