Cookbook—beadarrayMSV v1.0

Lars Gidskehaug*

September 2, 2010

1 Installation

- Install R (>= 2.10.0) (http://www.r-project.org/)
- Install Bioconductor packages "Biobase", "limma", and "geneplotter" (http://www.bioconductor.org/)
  ```
  > source("http://bioconductor.org/biocLite.R")
  > biocLite(c("Biobase","geneplotter","limma"))
  ```
- Install the interactive graphics package "GGobi" (http://www.ggobi.org/)
- Install packages "rggobi" and "beadarrayMSV" (http://cran.r-project.org/)
  ```
  > install.packages(c("rggobi","beadarrayMSV"))
  ```

2 Data-files

sampleFile Text-file (e.g. tab-separated) with extra information about the samples, containing at least the columns *Sample.ID*, *chip* (Sentrix ID), and *row* (Sentrix position)

beadFile Text-file (e.g. tab-separated) containing at least the columns *Name*, *SNP*, *Address*, *Address2*, *Norm.ID*, and *ILMN.Strand*

Output files from scanner Access to directory containing the output files of interest. Only the bead summary files are used, these have names such as '<chip>/ <chip>_<row>_<col>_beadTypeFile.txt' (*col* may be missing or have value 1)

3 Pre-processing

- Load package
  ```
  > library(beadarrayMSV)
  ```
- Load sample- and bead-info into data-frames
  ```
  > sampleInfo <- read.table(sampleFile,...)
  > beadInfo <- read.table(beadFile,...)
  ```

*CIGENE, N-1432 Ås; lars.gidskehaug@umb.no

- Load bead-summary data into a *BeadSetIllumina* object. This class has the required assay-data members *R*, *se.R*, *G*, *se.G*, and *no.beads*. The number of samples is often limited 100-200 samples at the time, depending on the assay and on computer specs

```
> BSDataRaw <- readBeadSummaryOutput(fullPaths=...
    fullPaths[1:step], ...)
> BSDataRaw
> dim(BSDataRaw)
> assayData(BSDataRaw)$G[1:10,1:4]
> pData(BSDataRaw)[1:10,]
> sampleNames(BSDataRaw)[1:10]
> varMetadata(BSDataRaw)
> fData(BSDataRaw)[1:10,]
> featureNames(BSDataRaw)[1:10]
> fvarMetadata(BSDataRaw)
> ...
```

- You may plot the signal for specific arrays

```
> scatterArrays(BSDataRaw,...)
```

- Make indexes to sub-bead pools

```
> normInd <- getNormInd(...)
```

- Set pre-processing options and constants. Defaults will be suggested

```
> normOpts <- setNormOptions()
> normOpts <- setNormOptions(transf='None',method='quantNorm')
```

$shearInf1 c([TRUE], FALSE) Controls whether all beads are used to find rotation/shearing-parameters or only Infinium I beads. The latter are supposed to have zero intensity in one of the channels and may therefore be well suited to find the baseline/origin

$minSize The signal along each channel is divided into bins, and a regression line is drawn through a (low) quantile of each bin in order to find the best rotation. Bins with fewer than *minSize* members are not included

$prob The quantile used to find regressor points for the homozygote asymptotes in the rotation/shearing

$nBins The number of bins used to find regressor points

$transf c('None', ['Root'], 'Log') The best transformation depends on the data. In general, non-transformed data are highly heteroscedastic, but they form clusters with little spread in *theta* (polar coordinates angle). A log-transformation tends to efficiently separate noise from signal, but it may lead to very imprecise values of *theta*. Some root-transformation is often a good, intermediate alternative

$nthRoot Which root, default: 4

$dist c(['euclidean'], 'manhattan', 'minkowski') The distance measure used when transforming from cartesian to polar coordinates. The best measure highly depends on the transformation used.

$pNorm The norm of the minkowski distance

$method c('none', 'quantNorm', ['medianAF'], 'linPeak') Controls which channel normalization to use in order to get comparable red and green signal

quantNorm Quantile normalization—forces the channels into similar distributions

medianAF Scales the red channel such that `median`$(\frac{R}{R+G}) \approx 0.5$

linPeak Scales each channel by its *scale*-th quantile

$scale Used with *linPeak* above

$nSD The number of standard deviations from the origin below which everything is regarded as noise

- See how the chosen pre-processing performs. Try different normOpts before you decide on which is best for your data
  ```
  > plotPreProcessing(BSDataRaw,...)
  ```
- Pre-process all the loaded samples
  ```
  > BSData <- preprocessBeadSet(BSDataRaw,...)
  ```
- Create an *AlleleSetIllumina* object. This class holds the marker-information, and has the required assay-data members *intensity*, *theta*, and *SE*. You may also wish to include additional, available sample information
  ```
  > BSRed <- createAlleleSet(BSData,beadInfo,normOpts)
  > pData(BSRed) <- combine(pData(BSRed),sampleInfo[1:step,])
  > BSRed
  > ...
  ```
- Unless the full data-set is small enough to be loaded into a single R-session, the pre-processed data must be written to files, and the rest of the data must be loaded and pre-processed in turn. Filenames may be generated automatically
  ```
  > tag <- '1stRun'
  > dataFiles <- makeFilenames(tag,normOpts,writePath)
  > writeAlleleSet(dataFiles[1:4],BSRed,append=FALSE)
  ```

  For subsequent genotype-calling, you may wish to try different clustering-options without needing to pre-process the data again. Then you may update the names of the result-files only
  ```
  > dataFiles <- modifyExistingFilenames(dataFiles,tag,'2ndRun')
  ```

  Write a for-loop in order to process the remaining samples and append the results to the data-files. You may at this point remove samples with mostly missing data if this is a problem.

4 Genotype calling

- Set genotype calling options and constants. Different defaults are suggested depending on whether you specify largeSample=TRUE (e.g 3000 samples) or FALSE (e.g. 100 samples)
  ```
  > gO <- setGenoOptions(largeSample=FALSE)
  > gO <- setGenoOptions(hwAlpha=1e-10,rotationLim=2)
  ```

$snpPerArrayLim Minimum ratio of non-NA markers per array

$arrayPerSnpLim Minimum ratio of non-NA arrays per marker

$ploidy c('di', 'tetra', 'tetra.red') Only *tetra* has been extensively tested

di Classifies markers into *MONO-a/-b*, or *SNP*

tetra Classifies markers into *MONO-a/-b*, *PSV*, *SNP*, or *MSV-a/-b/-5*

tetra.red Classifies markers into *MONO-a/-b*, *PSV*, *SNP*, or *MSV-a/-b*

$filterLim Markers with `range(theta)` below this value are filtered away (classified as *MONO-filt*). This is a quick way to discard homozygotes

$detectLim Markers with a ratio of called samples below this value is failed (classified as *FAIL*). Note there will always be a a few samples that are not called due to the way the cluster borders are estimated

$wSpreadLim The maximum allowed MAD along *theta* within a cluster

$devCentLim The maximum allowed deviation in *theta* between an 'ideal' and estimated cluster-center

$hwAlpha Significance level used in Hardy-Weinberg testing. Used to detect if clustering has failed, and a low value (e.g. 1e-10) shold be used to allow markers which deviates naturally from HW. This criterion should be used with caution, or set to zero, if e.g. the sample contains animals from different populations

$probsIndSE Optionally remove markers with standard errors above given quantile from clustering. May be set to 'NULL'

$afList At duplicated loci, the observed B allele-frequency (BAF) is in fact the mean BAF across both paralogues. Several values of BAF at paralogue 1, as given in $afList \in [0, 0.5]$, is tested, and the BAFs resulting in the best HW-significance are chosen

$clAlpha Significance level controlling the extent of an ellipse superimposed on each cluster, as estimated with a Hotelling's T^2-test. Animals falling outside the ellipse are not called, neither are animals within overlapping ellipses

$rPenalty Scaling-factor for the intensity axis. A value of 1 means the intensity is scaled with twice its median value, which gives approximately equal weight to *intensity* and *theta*. A higher value means the intensity is given less influence in the clustering (*theta* is relatively more important)

$rotationLim Controls the allowed angle of clusters, as defined by Hotelling's ellipses. A high value means horizontal clusters, a value of one means circular or angled 45 degrees, and a value close to zero means upright, well separated clusters. It is the weighted mean over clusters which is tested. Markers wrongly assigned as monomorphic often fail this test

$minClLim Clusters below this minimum size are disregarded in the Hotelling's test (all animals in cluster are included)

$nSdOverlap Number of standard deviations used in a cluster overlap criterion along *theta*. Two clusters overlap, resulting in a failed test, if the cluster-centers $\pm nSdOverlap \cdot wSpread$ overlap

$minBin The initial center-points are found from the peaks of a histogram of *theta*. The peaks with less than *minBin* samples are discarded as noise

$binWidth Histogram bin-width (controls smoothing/resolution)

- If the full data-set is too large for an R-session, load a specified number of markers from the files constructed in section 3
```
> BSRed <- createAlleleSetFromFiles(dataFiles[1:4],...)
```

- Perform automatic clustering and genotype calling
```
> BSRed <- callGenotypes(BSRed,g0=g0)
```

- If pedigree-information is available, it can be represented in a separate pData-column *PedigreeID*. This is in the format <p><mmm><fff><oo>, where 'p', 'mmm', 'fff' and 'oo' are unique identifiers for population, mother, father, and individual within full-sib group, respectively. '000' means founding parent, '999' means unknown. Pedigree-validation is a very powerful tool for assessing the quality of the genotyping and the clustering
```
> BSRed <- validateCallsPedigree(BSRed)
```

- Plot some markers to assess the calls. If pedigree-validation has been performed, violations will be given in yellow circles (parents) and crosses (offspring). Each cluster is given a unique colour depending on the B allele ratio, from red ($theta = 0$) to green ($theta = 1$)
```
> plotGenotypes(BSRed,1:72)
```

  Erroneous or strange calls should be scrutinized thoroughly at this point. There is a function which moves through all the tests during genotype calling, not failing any tests but returning the test outputs. This is a very important tool which can be used to tune the parameters optimally for your data. A scatter-plot for each attempted clustering displaying cluster centres (green) and overlapping (orange) or outlying (red) samples is displayed
```
> dev.new()
> verboseRes <- callGenotypes.verboseTest(BSRed,...)
> print(verboseRes$fData)
> print(verboseRes$test)
```

- You may discard samples which fail repeatedly at this point. NB! You should use a low value for the fail-ratio, as some samples fall just outside the cluster borders for many markers. Discarding many of these samples will introduce bias in the Hotelling's T^2-test during subsequent genotype calling
```
> BSRed <- countFailedSNP(BSRed,inclPedErrors=TRUE)
> indGoodArrays <- pData(BSRed)$passRatio > 0.2
> plotGenotypes(BSRed,1:72,which(!indGoodArrays))
```

- Loop through the full set of markers and save the results
```
> for (i in seq(1,nrow(beadInfo),mStep)){
>     ...
>     writeAlleleSet(dataFiles[5:6],BSRed,append=i>1)
>     ...
> }
```

- We recommend that you plot the genotypes for several markers within each classification in order to get a good overview of the performance of the calls. You may find that modification of *gO* is in order, as described above

- The calls are reported as the B allele ratios of the markers, $\frac{nB}{nA+nB}$. These may be translated into genotypes, either directly or from files
  ```
  > BSRed <- translateThetaCombined(BSRed)
  > translateThetaFromFiles(dataFiles,...)
  ```

5 Interactive clustering

- For erroneously called markers, or markers with bad cluster borders, there is some support for interactive clustering
  ```
  > BSManual <- callGenotypes.interactive(BSRed[indErr,],gO=gO)
  ```

- An *AlleleSetIllumina* object containing only the requested markers is produced. We recommend calling a limited number of markers at the time, as the procedure can be time-consuming and the results are not returned until all requested markers have been called. The manually updated calls are subsequently incorporated into the main *AlleleSetIllumina* object
  ```
  > BSRed <- assignToAlleleSet(BSRed,BSManual)
  ```

6 MSV-5 analysis

- When pedigree-information is available, and for samples with informative meioses, MSV-5's may be split into two individual paralogs and named accordingly. In the first step, this is performed for father and mother half-sib families independently
  ```
  > paraCalls <- unmixParalogues(BSRed[indMSV5,])
  ```

- For the following procedures, a linkage map is required. This is added to the feature-data of a *MultiSet* instance containing confidently called SNPs w.o. pedigree errors. (The *MultiSet* class of Bioconductor is similar to *BeadSetIllumina* and *AlleleSetIllumina*, however with no required assay-data members)
  ```
  > featureData(BSSnp) <- combine(featureData(BSSnp), lMap)
  ```
  The inherited allele from each parent is resolved where possible
  ```
  > inheritP <- resolveInheritanceSNP(BSSnp)
  ```

- The next step is to map each paralogue to a chromosome by counting matches to the SNPs in the map. A set of options is used to decide which comparisons count as a match. Default options are suggested
  ```
  > mergeOpts <- setMergeOptions()
  ```

  $minC The number of matches on a chromosome is divided by the number of SNPs mapped to the chromosome. Only the two largest peaks (chromosomes) are used, and only if they exceed *minC*

  $noiseQuantile If the above value is `NULL`, this quantile of the 3rd largest peaks across markers is used to estimate *minC*

$offspringLim The minimum number of offspring with equal inheritance between a paralogue and a mapped SNP

$ratioLim The minimum ratio of offspring with equal inheritance between a paralogue and a mapped SNP

$rngRF When searching for matches, also SNPs within $\pm rngRF$ map distance units of the mapped SNP is taken into account

- Count and plot the number of matches. The histograms often clearly reveal which chromosome(s) the paralog(s) belong to
  ```
  > chromHits <- locateParalogues(BSSnp,paraCalls,inheritP,...)
  > plotCountsChrom(chromHits$cPerMarker,1:72)
  ```

- The paralogs are named according to which chromosome they map to, and their (diploid) calls are returned. These may be translated into genotypes. A summary of the resolved paralogs may be arranged in a data-frame
  ```
  > mergedCalls <- assignParalogues(BSSnp,BSRed,...)
  > alleleCalls <- translateTheta(mergedCalls$x,...)
  > cCounts <- countChromosomes(...)
  ```

- A standalone mapping-software may be used to integrate the resolved paralogs into the linkage-map.