Identification of rare DNA variants using SRMA

Nianxiang Zhang & Wenyi Wang
Department of Bioinformatics and Computational Biology
Division of Quantitative Sciences, MDACC

January 6, 2012

1 Summary

It is still challenging to identify rare functional DNA variants. Custom array-based resequencing has an advantage of processing hundreds of individuals at reasonable cost and has been used for medical sequencing studies with targeted genes. However, high false positive rate (FPR ∼3%) in detection of heterozygous signals limited the application of this technology. We have previously developed novel statistical methodology for resequencing array data to largely reduce false discovery rate of single nucleotide variants (FPR$< 10^{-5}$, false negative rate $< 10\%$). We have established an analysis pipeline for Affymetrix resequencing microarray data analysis and assembled an R package named 'SRMA' (Sequence Robust Multi-array Analysis). The package contains five modules of quality control, data normalization, single array analysis, multi-array analysis, and output analysis. This tool allows processing of array-based sequencing data and identifying the footprint effects, insertions/deletions, and rare DNA single nucleotide variations (SNV) with high accuracy. We demonstrate the work flow of SRMA package in this manual.

2 Introduction

To understand the specific DNA variants that contribute to inheritance for human diseases is a major goal of human genetics. [1]. Medical resequencing has greatly accelerated the identification of disease related DNA variants. Resequencing array technology has been successfully used in identification of novel and potentially pathological variants in many genes involved in inherited diseases [2, 3]. However, the heterogeneity in data quality, low frequency of variant (1 in 1000 bp or lower), and spurious signals due to technical and experimental errors hinder the calling of rare variants. To overcome difficulties in detecting rare DNA variants, we have recently developed sequence robust multi-array analysis (SRMA) for resequencing array data analysis [4]. By improving pre-processing procedure, borrowing strength across samples and targeting unique features of rare variations, SRMA achieved a false discovery rate of 2% (FPR 1.2×10^{-5}, false negative rate (FNR) 5%), comparable to next-generation sequencing technologies. Here, we have established an R [5] package called 'SRMA' that fully implements these methods and provides an automated analysis pipeline for medical resequencing array data with high accuracy of calling rare variants.

3 System requirements

A system with a large memory capacity may be required for running the pipeline, depending on the design of chip type. For customized array "MitoP-2r520651", each .cel file is about 64 Mb. An 8G memory is necessary to process 50 .cel files. Therefore, a 64bit Windows or Linux/Unix system is required.

4 File structure requirements

In this package, the modules "quality control" and "data normalization" are contingent with the `aroma.affymetric` R package [6]. Therefore, a specific file structure compatible with aroma.affymetrix in the working directory should be set up before data processing. A CDF file and an ACS file corresponding to the chip type should be in sub-folder of 'annotationData' folder and .cel files should be in subfolder of 'rawData' folder.
The example dataset contains 37 .cel files located in folder "./rawData/example37cels/MitoP-2r520651". Where "example37cels" is the dataset name and "MitoP-2r520651" is the chip type name. The size of each .cel file is about 64Mb. The annotation files "MitoP-2r520651,byExon.CDF" and "MitoP-2r520651,WW20100206.acs" are in folder "./annotationData/chipTypes/MitoP-2r520651".
Besides the annotation files and raw data files, we also need two more datasets:

1. **map.rda**:A data frame, mapping between amplicon and exons. Structure of map is: row names are 1 to the total number of amplicons, column names are `exon, exon_len, exon_start, exon_end, amplicon, amp_len, amp_start, amp_end`
2. **ref.info.rda**:A dataframe containing columns of exon names (Fragment), position (FragPos) and reference allele (Ref) for each base in row.

The data structures of the two datasets are shown below:

```
> head(map, 3)

             exon exon_len exon_start exon_end        amplicon amp_len
1 ENSE00000333879      129          1      141 ENSE00000333879     230
2 ENSE00000343039       88          1      100 ENSE00000343039     193
3 ENSE00000343040       78          1       90 ENSE00000343040     187
  amp_start amp_end
1        49     189
2        54     153
3        38     127

> head(ref.info, 3)

         Fragment FragPos Ref
1 ENSE00001191811      13   t
2 ENSE00001191811      14   c
3 ENSE00001191811      15   t
```

5 Analysis

We will demonstrate analysis of resequencing data in the section. We load the `SRMA` package.

```
> require(SRMA)
```

5.1 Quality Control

The Quality control module is to identify the amplicons that are not suitable for base-calling due to failure in target amplification and hybridization, and is wrapped in function 'quality.control'. There are several steps in this module.

1. Raw intensities are read in using 'aroma.affymetrix' and normalized using strand-specific position normalization.
2. Three metrics including median of the average (log2) probe intensity, median of the log ratios, and reference call rate are calculated for each amplicon to evaluate the quality of the amplified targets.
3. A criterion of R<0.9 is used to identify failed amplicons.

The main inputs of the function 'quality.control()' are array related information and the data frame mapping amplicons to exons. The user can choose to calculate any combination of 3 metrics measuring quality of data. The outputs are a Boolean matrix indicating the failed amplicons in samples and the chosen quality control metrics.
We initiate the parameters/arguments and perform quality control. `aroma.affymetrix` file structure is required for this step. For the example data with 37 cel files, this step takes about 16 hours in a Linux system with dual hex-core Intel Xeon X5650, 2.6GHz processor and 16Gb memory.

```
> chipType <- "MitoP-2r520651"
> dataDir <- "example37cels"
> chipOrder <- 1:37
> output.measures <- c("D","T")
> mode.baselevel <- "average"
> mode.amplevel <- "average"
> exonList<-NA  #all exons

> qc.output <- quality.control(chipType = chipType, dataDir = dataDir,
+     chipOrder = chipOrder, output.measures = output.measures,
+     mode.baselevel = mode.baselevel, mode.amplevel = mode.amplevel,
+     map = map, exonList = exonList)
```

The structure of the outputs of quality control is shown below.

```
> str(qc.output, list.len = 8, vec.len = 2)

List of 5
 $       : NULL
 $ badamp: num [1:5620, 1:37] 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:5620] "AFFX-TagIQ-EX" "ENSE00000333879" ...
  .. ..$ : chr [1:37] "11840g_03_19_10 no1" "11840w_03_19_10 no6" ...
 $ D     : num [1:5620, 1:2, 1:37] 1.34 1.54 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ : chr [1:5620] "AFFX-TagIQ-EX" "ENSE00000333879" ...
  .. ..$ : chr [1:2] "sense" "antisense"
  .. ..$ : chr [1:37] "11840g_03_19_10 no1" "11840w_03_19_10 no6" ...
 $ T     : num [1:5620, 1:2, 1:37] 10.3 10.6 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ : chr [1:5620] "AFFX-TagIQ-EX" "ENSE00000333879" ...
  .. ..$ : chr [1:2] "sense" "antisense"
  .. ..$ : chr [1:37] "11840g_03_19_10 no1" "11840w_03_19_10 no6" ...
 $ par   :List of 4
```

```
  ..$ chipType : chr "MitoP-2r520651"
  ..$ dataDir  : chr "example37cels"
  ..$ chipOrder: int [1:37] 1 2 3 4 5 ...
  ..$ map      :'data.frame':        5619 obs. of  8 variables:
  .. ..$ exon      : Factor w/ 5471 levels "ENSE00000333879",..: 1 2 3 4 5 ...
  .. ..$ exon_len  : int [1:5619] 129 88 78 196 93 ...
  .. ..$ exon_start: num [1:5619] 1 1 1 1 1 ...
  .. ..$ exon_end  : num [1:5619] 141 100 90 208 105 ...
  .. ..$ amplicon  : Factor w/ 5619 levels "ENSE00000333879",..: 1 2 3 4 5 ...
  .. ..$ amp_len   : int [1:5619] 230 193 187 279 199 ...
  .. ..$ amp_start : int [1:5619] 49 54 38 36 56 ...
  .. ..$ amp_end   : int [1:5619] 189 153 127 243 160 ...
```

5.2 Normalization

Once we identify the failed amplicons, the intensities of the probes in those amplicons are changed to NAs and excluded from the normalization. Quantile normalization is performed at amplicon level. Then differences(δ) and averages(σ) between log2 transformed intensities of reference match (RM) probe and that of alternative match (AM) probes are calculated, the information of corresponding base pairs are recorded. The GC content and the length of exons are also calculated using the information of genotypes of reference alleles. The function 'get.delta.sigma.qn()' takes the outputs of quality control and performs normalization. The main outputs of the function are arrays of normalized data organized by bases, RM/AM basepairs, samples, and strands. We feed the result of QC module to the `get.delta.sigma.qn` to remove bad amplicon and do baselevel and amplicon level normalization.
In this step, a dataset that records the reference allele of each position(`ref.info`) is required. Again `aroma.affymetrix` file structure is required. This step takes about 15 hours.

```
> reorg.dat <- get.delta.sigma.qn(qc.out = qc.output, ref.dat = ref.info)
```

The structure of normalized data is shown below.

```
> str(reorg.dat, list.len = 6, vec.len = 2)
```

```
List of 6
 $ alldelta.qn: num [1:804113, 1:3, 1:37, 1:2] 1.67 2.35 ...
  ..- attr(*, "dimnames")=List of 4
  .. ..$ : chr [1:804113] "1" "2" ...
  .. ..$ : chr [1:3] "bp1" "bp2" ...
  .. ..$ : chr [1:37] "11840g_03_19_10 no1" "11840w_03_19_10 no6" ...
  .. ..$ : chr [1:2] "sense" "antisense"
 $ allsigma.qn: num [1:804113, 1:3, 1:37, 1:2] 10.7 10.3 ...
  ..- attr(*, "dimnames")=List of 4
  .. ..$ : chr [1:804113] "1" "2" ...
  .. ..$ : chr [1:3] "bp1" "bp2" ...
  .. ..$ : chr [1:37] "11840g_03_19_10 no1" "11840w_03_19_10 no6" ...
  .. ..$ : chr [1:2] "sense" "antisense"
 $ allpairs.qn: chr [1:804113, 1:3, 1:2] "TA" "GA" ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ : chr [1:804113] "1" "2" ...
  .. ..$ : chr [1:3] "bp1" "bp2" ...
```

```
  .. ..$ : chr [1:2] "sense" "antisense"
 $ ref.info   :'data.frame':        804113 obs. of  3 variables:
  ..$ Fragment: Factor w/ 6030 levels "AFFX-TagIQ-EX",..: 1 1 1 1 1 ...
  ..$ FragPos : int [1:804113] 175 176 177 178 179 ...
  ..$ Ref     : Factor w/ 4 levels "a","c","g","t": 1 2 1 3 2 ...
 $ gc         : num [1:5472] 0.506 0.41 ...
 $ exonlength : int [1:5472] 814 139 118 186 153 ...
```

5.3 Single Array Analysis

In single array analysis module, for each sample, a linear model is used to adjust the log ratio δ's for a set of explainable variables including average intensity σ, amplicon length, amplicon GC content and central base pair composition. For σ, amplicon length, and amplicon GC content, piecewise cubic spline regression bases [7] are used in the linear model as covariates. We assume a Gaussian distribution for adjusted δ given the allele variant class, identical and independent distributions for each strand, and the prior probabilities for the three variant classes RR, RS, and SS being (0.998, 0.002, 0.000001). The posterior probabilities of RR, RS and SS genotype are calculated for all three choices of alternative alleles in each nucleotide position. Then the alternative alleles that maximize the expected number of homozygous variants SS are chosen using these posterior probabilities.
The subset of data for the chosen alternative alleles is retrieved from the original normalized data. Log ratios are readjusted using the similar linear model described above except for that the weighted least squares method is used due to the presence of heteroscedasticity [4]. The weights are estimated using R package `nlme` [8] . The single array posterior probabilities for the three variant classes are recalculated using adjusted data.
The main input of the function "single.chip.posterior.probability()" is an object from data normalization module that includes the normalized data and information of each nucleotide position. The outputs of the function are the indices of chosen alternative alleles, the posterior probabilities of three variant classes, and the parameters from linear models used to adjust the logratios.
single chip analysis step takes about 38 hours for our 37 samples.

```
> sd.adj.dat <- single.chip.posterior.prob(reorg.dat)

> str(sd.adj.dat, list.len = 6, vec.len = 3)

List of 3
 $ post.prob :List of 5472
  ..$ : num [1:814, 1:37, 1:3] 1 1 0.995 1 ...
  ..$ : num [1:139, 1:37, 1:3] 1 1 1 1 ...
  ..$ : num [1:118, 1:37, 1:3] 1 1 1 1 ...
  ..$ : num [1:186, 1:37, 1:3] 1 1 1 1 ...
  ..$ : num [1:153, 1:37, 1:3] NA NA NA NA NA NA NA NA ...
  ..$ : num [1:113, 1:37, 1:3] 1 1 1 1 ...
  .. [list output truncated]
 $ allbp.indx: int [1:804113] 3 3 1 1 3 1 1 1 ...
 $ chipadj.qn:List of 6
  ..$ f1     : num [1:804113, 1:37, 1:2] 0.734 1.92 1.31 1.091 ...
  ..$ f3     : num [1:804113, 1:37, 1:2] -1.746 -0.908 -1.24 -1.741 ...
  ..$ mu1    : num [1:37, 1:2] 1.77 1.73 1.76 1.86 ...
  ..$ mu3    : num [1:37, 1:2] -1.73 -1.68 -1.72 -1.79 ...
```

```
  ..$ errors1: num [1:804113, 1:37, 1:2] 0.269 0.496 0.356 0.318 ...
  ..$ errors3: num [1:804113, 1:37, 1:2] 0.451 0.29 0.343 0.45 ...
```

Multiarray analysis starts from initial genotypes assignment based on single array posterior probability. For the positions where all samples were designated as RR with the corresponding posterior probabilities >0.999, they are reference-only positions. For the other positions, k-means clustering are used to designate preliminary genotypes; the minor allele count (MAC) defined as the total number of alternative alleles across all samples for a nucleotide position is calculated; and a Gaussian mixture model is used to genotype all samples based on the logratio δ's. However, different approaches are used according to the MAC.
For common variants position with MAC>=4, EM algorithms implemented in R package mclust [9] are used to obtained maximum likelihood estimates for the model parameters (m for the centers and v for the covariance matrices)and Bayesian information criterion (BIC) is used to determined the number of clusters. For rare variant positions with MAC<4, the multi-array posterior probabilities are calculated based on the following assumptions:

1. m for the heterozygous cluster is (0,0)
2. m for the homozygous variant cluster is the minus of the average values of mean for observed homozygous reference cluster across all bases
3. Covariance matrices for the non-reference clusters are identical to the corresponding reference cluster at the same base positions
4. Logratios follow bivariate Gaussian distribution for the two strands given the nucleotide position and the genotype of the position
5. The prior probabilities for 3 genotype classes are (0.998, 0.002, 0.000001) for unknown variant positions and (0.98, 0.02, 0.0001) for dbSNP positions.

The genotype class with the highest posterior probability among all classes is assigned to each position for each sample. A quality score measuring Silhouette width [10] and a reliability score measuring signal to noise ratio are calculated to assess the quality of the clustering.
The function 'srma()' performs multiarray analysis on a amplicon-by-amplicon basis. The main inputs of the function are the single array posterior probabilities, alternative allele indices, normalized data, and list of dbSNPs. The main outputs of the function are the genotype calls, multiarray posterior probabilities, quality scores and reliability scores.

```
> srma.out <- call.genotype(reorg.dat, sd.adj.dat)
```

The structure of output from multiarray analysis is shown below. This step takes about 17 hours to complete for our 37 samples.

```
> str(srma.out, list.len = 3, vec.len = 2)
```

```
List of 3
 $ output      :List of 5472
  ..$ AFFX-TagIQ-EX  :List of 5
  .. ..$ calls     : chr [1:814, 1:37] "AA" "CC" ...
  .. ..$ posteriors: num [1:814, 1:37] 1 1 ...
  .. ..$ category  : chr [1:814] "rare variants" "rare variants" ...
  .. .. [list output truncated]
  ..$ ENSE00000925350:List of 5
```

```
  .. ..$ calls     : chr [1:139, 1:37] "TT" "GG" ...
  .. ..$ posteriors: num [1:139, 1:37] 1 1 1 1 1 ...
  .. ..$ category  : chr [1:139] "ref only" "ref only" ...
  .. .. [list output truncated]
  ..$ ENSE00000925351:List of 5
  .. ..$ calls     : chr [1:118, 1:37] "CC" "TT" ...
  .. ..$ posteriors: num [1:118, 1:37] 1 1 1 1 1 ...
  .. ..$ category  : chr [1:118] "ref only" "ref only" ...
  .. .. [list output truncated]
  .. [list output truncated]
 $ sample.names: chr [1:37] "11840g_03_19_10 no1" "11840w_03_19_10 no6" ...
 $ refCalls    :'data.frame':        804113 obs. of  3 variables:
  ..$ Fragment: Factor w/ 6030 levels "AFFX-TagIQ-EX",..: 1 1 1 1 1 ...
  ..$ FragPos : int [1:804113] 175 176 177 178 179 ...
  ..$ Ref     : Factor w/ 4 levels "a","c","g","t": 1 2 1 3 2 ...
```

5.4 Output analysis

Output analysis includes detection of technical artefacts and identification of reliable rare SNVs and insertion/deletions. To detect and eliminate the heterozygous call from footprint effect artefacts, low-homology regions, PCR errors or local array defects, rare heterozygous calls (MAC<4) in the flanking 24 bp of a homozygous base-call for the same sample are removed if no other homozygous calls are made at those positions, and more than two heterozygous calls (MAC<4) in the same amplicon of one sample are removed. Then small insertion/deletions are detected by determine whether a string of heterozygous calls in a sample is present. In order to maintain a reasonable balance between FDR and FNR, a default quality score of 0.67 is used to exclude the low quality genotype calls based on our validation data [4].
The function "post.processing()" takes the output of multi-array analysis and generates two tab-delimited tables containing bases with insertion/deletions and SNVs and genotype calls for those bases in each sample. This step takes 40 minute to complete.

```
> Results <- post.process(srma.out)
```

We can examine the summarized analysis results.

```
> summary(Results)
```

```
Analysis Results From SRMA Analysis Pipeline:
------------------------------------------------------
------------------------------------------------------
 QC:
               Total QC.Failed Percent.Failed
Base Number 29752181   5853915       19.67558
------------------------------------------------------
 Footprint Effects:
     Bases: 4930
     Samples: 1105
     0.0166 percent of bases are affected by footprint effects.
------------------------------------------------------
 Silhouette Scores:
     Summary of Silhouette scores (cutoff= 0.67 ).
```

```
               Low    Total   Percent
Base Number 5552974 23898266 0.2323589
-----------------------------------------------------
 Single nucleotide variations:
      3096 SNP positions, 104719 SNPs are called in 37 samples.
     0.4382 percent of variant calls are SNPs.
-----------------------------------------------------
 Insertion/Deletions:
      2451 insertion/deletions are identified.
     Insertion/deletions are identified in 457 exons.
```

Finally, we can save the SNP calls and indel calls.

```
> write.table(Results@srmaSnps, file = "SnpCalls.txt")
> write.table(Results@srmaIndelsnps, file = "IndelCalls.txt")
```

6 Environment

```
> getwd()
```

```
[1] "/workspace/NZhang/WWang101210/SRMA/doc"
```

```
> sessionInfo()
```

```
R version 2.12.0 (2010-10-15)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=C              LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] splines   stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] SRMA_1.0.0-2           ellipse_0.3-5          mnormt_1.4-0
 [4] mclust_3.4.7           MASS_7.3-8             aroma.affymetrix_1.7.0
 [7] aroma.apd_0.1.7        affxparser_1.22.0      R.huge_0.2.0
[10] aroma.core_1.7.0       aroma.light_1.18.2     matrixStats_0.2.2
[13] R.rsp_0.4.0            R.cache_0.3.0          nlme_3.1-97
[16] R.filesets_0.9.0       digest_0.4.2           R.utils_1.5.3
[19] R.oo_1.7.4             R.methodsS3_1.2.1

loaded via a namespace (and not attached):
[1] grid_2.12.0     lattice_0.19-13 tools_2.12.0
```

References

[1] Bodmer W, Bonilla C. (2008) Common and rare variants in multifactorial susceptibility to common diseases. *Nat Genet.*, **40(6)**, 695-701.

[2] Rubio JP, Wilkins EJ, Kostchet K, Cowie TC, O'Hely M, Burfoot R, Wang W, Speed TP, Stankovich J, Horne M.(2011) A DNA Resequencing Array for Genes Involved in Parkinson's Disease. *Parkinsonism & Related Disorders* **accepted**

[3] Shen, P. *et al.* (2011) High-quality DNA sequence capture of 524 disease candidate genes. *PNAS*, **108**, 6549-6554.

[4] Wang, W. *et al.* (2011) Identification of rare DNA variants in mitochondrial disorders with improved array-based sequencing. *Nucleic Acids Res*, **39**, 44-58.

[5] R Development Core Team (2010) R: A language and environment for statistical computing., *R Foundation for Statistical Computing*, Vienna, Austria. ISBN 3-900051-07-0.

[6] Bengtsson, H. *et al.* (2008) aroma.affymetrix: A generic framework in R for analyzing small to very large Affymetrix data sets in bounded memory. Tech Report #745, Department of Statistics, University of California, Berkeley.

[7] Fritsch, F. N. and Carlson, R. E. (1980) Monotone piecewise cubic interpolation, *SIAM Journal on Numerical Analysis*, **17**, 238-246.

[8] Pinheiro, J.C. and Bates, D.M. (2000) Mixed-Effects Models in S and S-PLUS, *Springer*.

[9] Fraley, C. and Raftery, AE. (2002) Model-based clustering, discriminant analysis, and density estimation. *JASA*, **97**, 611-631.

[10] Rousseeuw PJ. (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Mathematics*, **20**,53-65.