MouseDivGeno Vignette (Version 1.0.3)

Hyuna Yang and Keith Sheppard

December 14, 2011

1 Introduction

`MouseDivGeno` is a R package specifically designed to genotype the Mouse Diversity Genotyping Array (Yang et al, 2008), an Affymetrix mouse genotyping array similar to the human SNP 6.0. `MouseDivGeno` contains functions which allow you to perform genotyping, identify probe sets potentially harboring a new mutation (Variable INtensity Oligonucleotide or VINO, here we call it a vinotyping) and perform CNV analysis. The R package, annotations, and further information can be obtained from http://genomedynamics.org/tools/mousedivgeno.

2 Installation

Before installing this package you should install the package dependencies:

- required packages: cluster, affyio, preprocessCore
- optional packages: HiddenMarkov, multicore

You can use the following commands to install all of the dependencies:

```
> # install packages from cran
> install.packages("cluster")
> install.packages("HiddenMarkov")
> install.packages("multicore")
> # install bioconductor packages
> source("http://bioconductor.org/biocLite.R")
> biocLite("affyio")
> biocLite("preprocessCore")
```

Now that you have installed the dependencies you can install the MouseDivGeno package. On Mac OS X or Linux you can do this by entering the following command in your terminal:

```
> R CMD INSTALL MouseDivGeno_1.0.3.tar.gz
```

If you are using windows start R and use "packages->Install package(s) from local zip files..." to install `MouseDivGeno_1.0.3.tar.gz`

3 Quality Checks for the Mouse Diversity Genotyping Array

Before we get started we will to load some data into scope which will give us probe and probeset level information about the Mouse Diversity Array platform which we will use in several examples throughout this vignette. You can obtain this data from http://genomedynamics.org/tools/mousedivgeno

```
> load(file.path(working_dir, "MouseDivData.RData"))
> ls()
```

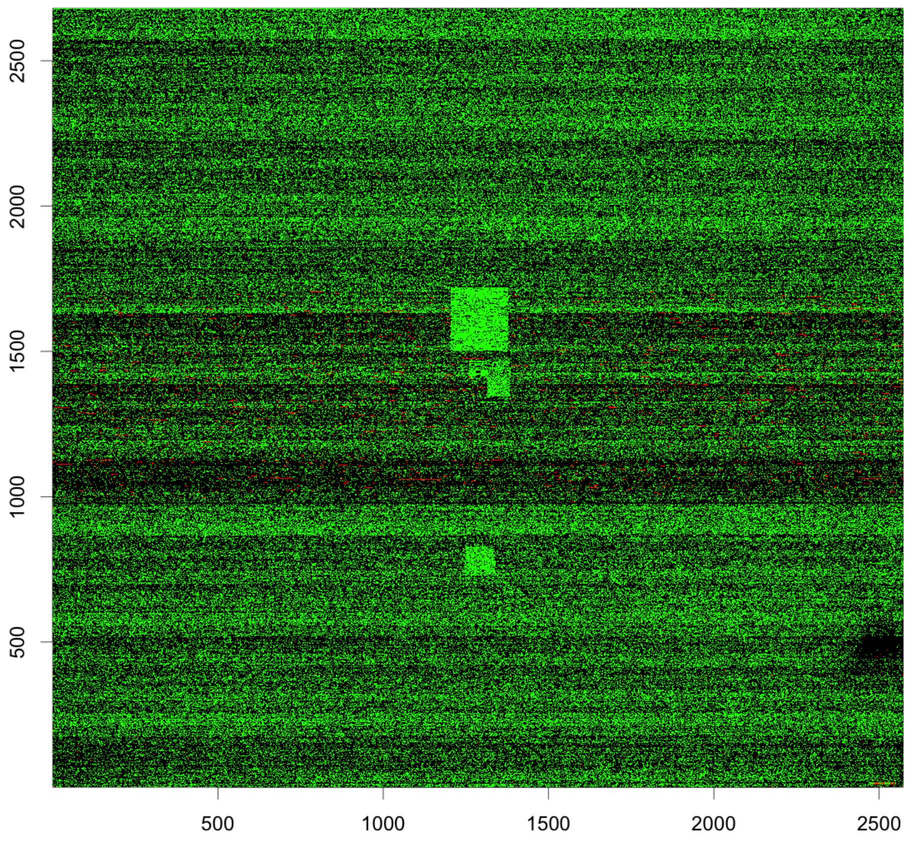

Figure 1: Array Image from plotMouseDivArrayImage(...)

```
[1] "invariantProbeInfo"
[2] "invariantProbesetInfo"
[3] "invariantReferenceDistribution"
[4] "snpInfo"
[5] "snpProbeInfo"
[6] "snpReferenceDistribution"
[7] "working_dir"
```

You will want to perform some quality checks on the arrays that you are genotyping. The quality checks in this section are specific to the Mouse Diversity Genotyping Array. First we can plot an image of an array to look for any obvious problems. This will create log2 intensity heatmap and will allow you to determine if the array has a spatial distribution:

```
> plotMouseDivArrayImage(file.path(working_dir, "celFiles/SNP_mDIV_A8-8_081308.CEL"))
```

Note that because we use four probes (two from the sense strand and two from the antisense strand) per probe set and those probes are randomly located across .CEL file, if there is a dim region, it does not affect the overall intensity due to the median summarization step. However if some .CEL files show an unusually big dark spot, or overall dark image, check the array processing steps including reagent, scanner, etc

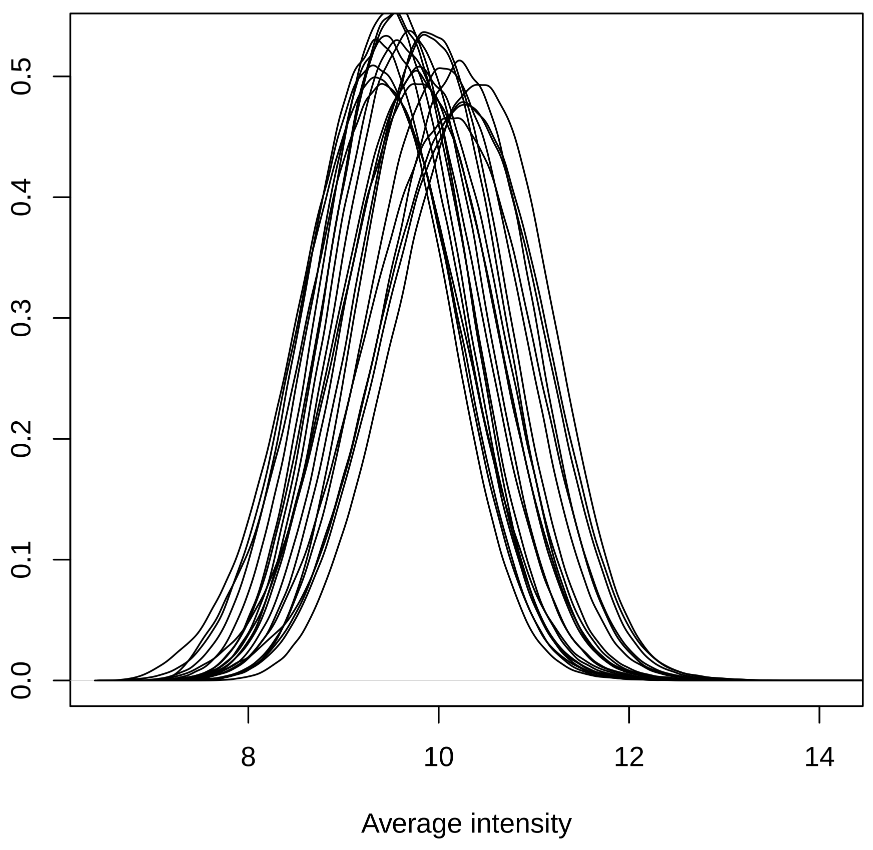

Figure 2: Average SNP Intensities: (A + B) / 2

We can also compare the set of arrays that we plan to genotype against eachother by doing a density plot of thier SNP intensities (each plot line represents a different array). This can help you to determine whether your arrays should be quantile normalized or not:

```
> mouseDivDensityPlot(file.path(working_dir, "celFiles"), snpProbeInfo)
```

Likewise we can do a boxplot of the A and B allele intensities of our CEL files. In this example we will plot the intensities of `SNP_mDIV_A1-1_081308.CEL` and `SNP_mDIV_A10-10_081308.CEL`.

```
> celInten <- readCELFiles(
+     c(
+         file.path(working_dir, "celFiles/SNP_mDIV_A1-1_081308.CEL"),
+         file.path(working_dir, "celFiles/SNP_mDIV_A10-10_081308.CEL")),
+     snpProbeInfo)
> boxplot(celInten)
```

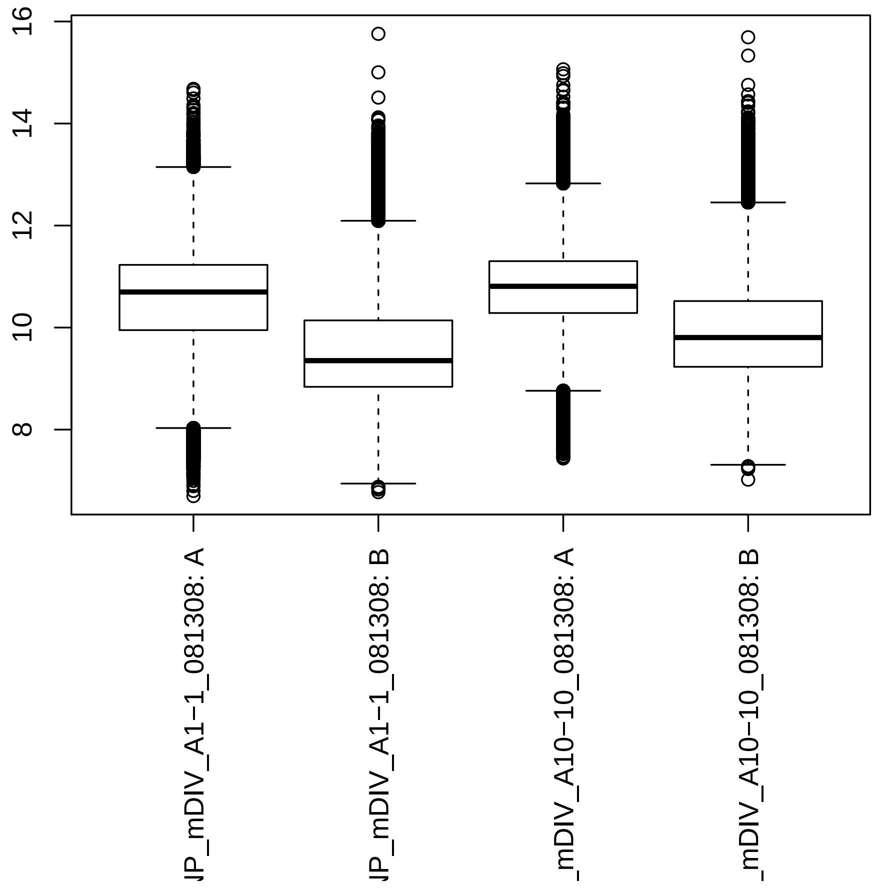

Figure 3: Boxplot the intensities of two CEL files

4 Genotyping and Vinotyping for the Mouse Diversity Genotyping Array

For the purposes of genotyping we only care about the ”snp” data so we will take a close look at these objects.

- `snpProbeInfo`: this data frame contains all of the probe-level information about snp probes in the mouse diversity platform:

  ```
  > snpProbeInfo[1 : 2, ]

    probeIndex isAAllele       snpId correction
  1          6     FALSE JAX00194170  0.3034845
  2          7      TRUE JAX00194170  0.3034845
  ```

  Column descriptions:

  – `probeIndex`: the CEL file index for the probe (used to get intensity data)

  – `isAAllele`: `TRUE` for A allele probes and `FALSE` for B allele probes

  – `snpId`: the ID of the SNP that this probe belongs to. These IDs should correspond to the same component in the `snpInfo` argument

  – `correction` (optional): if present this correction will be applied by adding it to probe mean intensity vales. In the MouseDivData object that we have provided these values have been calculated to correct intensity variation due to C G content in 25mers and restriction fragment size. These coefficients were calculated using a spline regression model fitting

- `snpInfo`: this data frame contains SNP-level information:

  ```
  > snpInfo[1 : 2, ]

                    snpId chrId positionBp isInPAR snpHetHint
  JAX00000001 JAX00000001     1    3013441   FALSE -0.2205501
  JAX00000002 JAX00000002     1    3036178   FALSE -0.2517160
              flagged alleleA alleleB   rsNumber
  JAX00000001   FALSE       A       T rs31192577
  JAX00000002   FALSE       A       C rs32166183
              ensemblTranscriptID ensemblGeneID
  JAX00000001                 N/A           N/A
  JAX00000002                 N/A           N/A
              substitutionType locusSymbol functionClass
  JAX00000001     Transversion         N/A    Intergenic
  JAX00000002     Transversion         N/A    Intergenic
  ```

  Column descriptions:

  – `snpId`: the ID of this SNP

  – `chrId`: the chromosome that this SNP belongs to. Like ”15” or ”X”

  – `snpHetHint` (optional): provides an initial hint value for the normalized mean intensity of heterozygous SNPs. In the MouseDivData object that we have provided these hints were calculated based on 350 training arrays.

  – `isPAR` (optional): if `TRUE` this indicates that this SNP is in the pseudoautosomal region (PAR). This should only be set to `TRUE` for SNPs where `snpInfo$chrId == "X"`

  – `...` (optional): all of the remaining columns are ignored by the `MouseDivGeno` package and are only present as a convenience in order to help you to annotate your results

- `snpReferenceDistribution`: a numeric vector which is the reference distribution used for the quantile normalization of SNP intensities In the MouseDivData object that we have provided these normalized coefficients were calculated based on 350 training arrays.

Now we're ready to genotype our CEL files. To do this we will use the `mouseDivGenotypeCEL(...)` function. You will notice in this example that we use all of the SNP data objects described above. We also are only genotyping four chromosomes rather than the default which is to genotype all chromosomes. As specified here gender will be inferred from the CEL files but you can also explicitly set gender if you have that information ahead of time. See the `mouseDivGenotypeCEL(...)` function's documentation for details on how to do this.

```
> # we use confScoreThreshold = 0 so that we get most likely genotypes even for
> # the low confidence SNPs. We will need these genotypes later for CNV analysis
> genoVinoResult <- mouseDivGenotypeCEL(
+     snpProbeInfo             = snpProbeInfo,
+     snpInfo                  = snpInfo,
+     referenceDistribution    = snpReferenceDistribution,
+     chromosomes              = c(19, "X", "Y", "M"),
+     celFiles                 = file.path(working_dir, "celFiles"),
+     confScoreThreshold       = 0)
```

The `genoVinoResult` object contains the genotype calls, vinotype calls and confidence values for every SNP along with an isMale vector which indicates for each CEL file whether the sample was inferred as male or female (the `mouseDivGenotypeCEL(...)` documentation shows how you can provide gender information using the `celFiles` argument if you know it a priori). Here is a small subset of the output to give you an idea of what it looks like:

```
> genoVinoResult$geno[1 : 2, 10 : 11]

            SNP_mDIV_A8-8_081308 SNP_mDIV_A9-57_082108
JAX00086300                    1                     1
JAX00086302                    3                     1

> genoVinoResult$vino[1 : 2, 10 : 11]

            SNP_mDIV_A8-8_081308 SNP_mDIV_A9-57_082108
JAX00086300                    0                     0
JAX00086302                    0                     0

> genoVinoResult$conf[1 : 2, 10 : 11]

            SNP_mDIV_A8-8_081308 SNP_mDIV_A9-57_082108
JAX00086300            0.8096121             0.9830769
JAX00086302            0.9396675             0.1406445

> genoVinoResult$isMale[10 : 11]

 SNP_mDIV_A8-8_081308 SNP_mDIV_A9-57_082108
                 TRUE                 FALSE
```

You can interpret the returned matrix values as follows:

- `genoVinoResult$geno`: -1 = No call, 1 = AA, 2 = AB, and 3 = BB
- `genoVinoResult$vino`: 1 indicates VINO, 0 indicates no VINO
- `genoVinoResult$conf`: These values are confidence scores ranging from 0 to 1 based on Mohalanobis distance with chi-squre distribution approximation. Smaller confidence score implies less reliable data.

Now if we want we can annotate our results to make them a little bit more meaningful.

```
> annoResults <- matrix("", ncol=ncol(genoVinoResult), nrow=nrow(genoVinoResult))
> rownames(annoResults) <- rownames(genoVinoResult)
> colnames(annoResults) <- colnames(genoVinoResult)
> annoResults[c(genoVinoResult$geno) == -1] <- "N"
> annoResults[c(genoVinoResult$geno) == 1] <- "A"
> annoResults[c(genoVinoResult$geno) == 2] <- "H"
> annoResults[c(genoVinoResult$geno) == 3] <- "B"
> annoResults[c(genoVinoResult$vino) == 1] <- "V"
> infoMatched <- snpInfo[match(rownames(genoVinoResult), snpInfo$snpId), ]
> annoResults <- cbind(infoMatched, annoResults)
> # now let's just take a look at just one of the annotated SNP results
> annoResults[2, , drop=FALSE]
```

```
                  snpId chrId positionBp isInPAR snpHetHint
JAX00086302 JAX00086302    19    3159638   FALSE -0.1117551
            flagged alleleA alleleB   rsNumber
JAX00086302   FALSE       T       G rs49368223
            ensemblTranscriptID ensemblGeneID
JAX00086302                 N/A           N/A
            substitutionType locusSymbol functionClass
JAX00086302     Transversion         N/A    Intergenic
            SNP_mDIV_A1-1_081308 SNP_mDIV_A10-10_081308
JAX00086302                    A                      A
            SNP_mDIV_A11-11_081308 SNP_mDIV_A2-2_081308
JAX00086302                      A                    A
            SNP_mDIV_A3-3_081308 SNP_mDIV_A4-4_081308
JAX00086302                    A                    A
            SNP_mDIV_A5-5_081308 SNP_mDIV_A6-6_081308
JAX00086302                    A                    A
            SNP_mDIV_A7-7_081308 SNP_mDIV_A8-8_081308
JAX00086302                    A                    B
            SNP_mDIV_A9-57_082108 SNP_mDIV_A9-9_081308
JAX00086302                     A                    B
            SNP_mDIV_B1-12_081308 SNP_mDIV_B2-13_081308
JAX00086302                     H                     H
            SNP_mDIV_B3-14_081308 SNP_mDIV_B4-15_081308
JAX00086302                     A                     A
            SNP_mDIV_B5-16_081308 SNP_mDIV_B6-17_081308
JAX00086302                     B                     B
            SNP_mDIV_B7-18_081308 SNP_mDIV_D10-187_082108
JAX00086302                     A                       H
            SNP_mDIV_D3-179_082108
JAX00086302                      A
```

4.1 Subsetting the SNPs To Genotype

As already demonstrated by the example code for `mouseDivGenotypeCEL(...)` you can use the `chromosomes` parameter to easily subset the genotyped SNPs by chromosome. If you need a more fine grained control for excluding SNP genotypes from your results you can use one of the following two strategies:

Genotype as usual and subset the results as a post processing step. For instance, if we want to look at just the SNPs between 60,000,000 and 60,030,000 mega base pairs on chromosome 19 we can do something like this.

```
> # figure out which IDs fall between 60,000,000 and 60,030,000
> snpsToKeep <- snpInfo$chrId == "19" & snpInfo$positionBp >= 60000000 & snpInfo$positionBp <= 60030000
> snpIdsToKeep <- snpInfo$snpId[snpsToKeep]
> snpIdsToKeep

[1] "JAX00481784" "JAX00481785" "JAX00481786"

> # subset the data based on the IDs that we've found
> genoVinoResultSubset <- genoVinoResult[match(snpIdsToKeep, rownames(genoVinoResult)), ]
> genoVinoResultSubset$geno[, 3 : 4]

            SNP_mDIV_A11-11_081308 SNP_mDIV_A2-2_081308
JAX00481784                      1                    1
JAX00481785                      1                    1
JAX00481786                      2                    3

> genoVinoResultSubset$vino[, 3 : 4]

            SNP_mDIV_A11-11_081308 SNP_mDIV_A2-2_081308
JAX00481784                      0                    0
JAX00481785                      0                    0
JAX00481786                      0                    0

> genoVinoResultSubset$conf[, 3 : 4]

            SNP_mDIV_A11-11_081308 SNP_mDIV_A2-2_081308
JAX00481784              0.6618886            0.9852621
JAX00481785              0.8433786            0.9236106
JAX00481786              0.8393652            0.7071218
```

Alternatively you can subset the SNP data that you give to the `genotypeSnps(...)` function as shown in the example below.

```
> celDataSubset <- readCELFiles(
+     celFiles               = file.path(working_dir, "celFiles"),
+     snpProbeInfo           = snpProbeInfo,
+     referenceDistribution  = snpReferenceDistribution,
+     snpIdsToKeep           = snpIdsToKeep)
> celDataSubset <- ccsTransform(celDataSubset)
> genoVinoResultSubset <- genotypeSnps(celDataSubset, snpInfo)
```

This approach can be much faster than subseting all of the genotypes after genotyping but you do have to be careful if you are genotyping on the X or Y chromosomes. If you are genotyping X or Y SNPs and you do not provide gender information via the `isMale` parameter you must include representative SNPs from at least one of the autosomes, the X chromosome and the Y chromosome. This is because the algorithm for genotyping X or Y SNPs relies on sample gender which can only be inferred if representative autosome, X and Y data are present.

4.2 Genotyping with Few Samples

The genotyping algorithms used by `MouseDivGeno` requires a large set of samples, such as the set of 351 CEL files which are made available at http://genomedynamics.org/tools/mousedivgeno, in order to generate high quality genotype results. However you may only be interested in obtaining the genotypes for a small subset of these samples. You can of course do this by genotyping all of the samples as shown in previous examples and subsetting the results after the fact, but it will be much more memory efficient to use the `samplesToKeep` parameter as shown in the example below:

```
> samplesToKeep <- c("SNP_mDIV_A8-8_081308", "SNP_mDIV_A9-57_082108")
> genoVinoResultKeep2 <- mouseDivGenotypeCEL(
+     snpProbeInfo            = snpProbeInfo,
+     snpInfo                 = snpInfo,
+     referenceDistribution   = snpReferenceDistribution,
+     samplesToKeep           = samplesToKeep,
+     chromosomes             = c(19, "X", "Y", "M"),
+     celFiles                = file.path(working_dir, "celFiles"),
+     confScoreThreshold      = 0)
> genoVinoResultKeep2$geno[1 : 2, ]

            SNP_mDIV_A8-8_081308 SNP_mDIV_A9-57_082108
JAX00086300                    1                     1
JAX00086302                    3                     1

> genoVinoResultKeep2$vino[1 : 2, ]

            SNP_mDIV_A8-8_081308 SNP_mDIV_A9-57_082108
JAX00086300                    0                     0
JAX00086302                    0                     0

> genoVinoResultKeep2$conf[1 : 2, ]

            SNP_mDIV_A8-8_081308 SNP_mDIV_A9-57_082108
JAX00086300            0.8096121             0.9830769
JAX00086302            0.9396675             0.1406445

> genoVinoResultKeep2$isMale

 SNP_mDIV_A8-8_081308 SNP_mDIV_A9-57_082108
                 TRUE                 FALSE
```

5 Genotyping and Vinotyping on Other Platforms

As described in the previous section you can directly genotype your CEL files if you are using the Mouse Diversity Genotyping Array with a single function call, but if you happen to be using a different genotyping platform you will need to first generate per-SNP A-allele/B-allele intensity values before you will be able to use this package. Here we assume those intensity matrices are named `aIntensities` and `bIntensities`:

```
> dim(aIntensities)

[1] 16179    21

> dim(bIntensities)

[1] 16179    21
```

Now we can easily turn these intensities into the contrasts and average terms and genotype them:

```
> # for this example I will only genotype SNPs 15 and 1281
> snpInten <- makeSnpIntensities(
+     aIntensities[c(15, 1281), ],
+     bIntensities[c(15, 1281), ],
+     snpIds=c("SNP15", "SNP1281"),
+     sampleNames=colnames(aIntensities))
> snpContAvg <- ccsTransform(snpInten)
> # Here we create a minimal snpInfo data frame that contains
> # SNP IDs and chromosome IDs for each SNP
> minimalSnpInfo <- data.frame(
+     snpId=c("SNP15", "SNP1281"),
+     chrId=c("19", "19"))
> # finally we can genotype the SNPs and print the result
> genoVinoConf <- genotypeSnps(snpContAvg, minimalSnpInfo)
> genoVinoConf

$geno
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    1    1    3    3    1    1    3    1     1
[2,]    1    1    1   -1   -1    3    1    3    1     3
     [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19]
[1,]     1     1     1     3     3     1     1     1     1
[2,]     3     3    -1    -1    -1     3     3     3     3
     [,20] [,21]
[1,]     2     2
[2,]     3     1

$vino
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    0    0    0    0    0    0    0    0    0     0
[2,]    0    1    0    1    1    0    0    0    0     0
     [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19]
[1,]     0     0     0     0     0     0     0     0     0
[2,]     1     0     1     1     1     0     0     0     0
     [,20] [,21]
[1,]     0     0
[2,]     1     0

$conf
          [,1]      [,2]      [,3]      [,4]         [,5]
[1,] 0.1705322 0.6859672 0.7028156 0.9134067 8.848209e-01
[2,] 0.7802313 0.3488805 0.9903111 0.0000000 1.202033e-10
          [,6]      [,7]      [,8]      [,9]     [,10]
[1,] 0.5750916 0.5420340 0.8853103 0.9642561 0.9918969
[2,] 0.9943182 0.5601468 0.6765889 0.7768043 0.8855664
         [,11]     [,12]        [,13]     [,14]
[1,] 0.5174464 0.4651205 2.295647e-01 0.2495163
[2,] 0.5285513 0.7442445 5.133813e-08 0.0000000
            [,15]     [,16]     [,17]     [,18]     [,19]
[1,] 2.407099e-01 0.9128792 0.5457933 0.6526228 0.8954189
[2,] 1.110223e-16 0.9938091 0.7550623 0.8119089 0.9418095
           [,20]     [,21]
[1,] 0.517149505 0.5171495
[2,] 0.000792704 0.9903111

$probesetIds
[1] "SNP15"   "SNP1281"

$sampleNames
 [1] "SNP_mDIV_A1-1_081308"    "SNP_mDIV_A10-10_081308"
 [3] "SNP_mDIV_A11-11_081308"  "SNP_mDIV_A2-2_081308"
 [5] "SNP_mDIV_A3-3_081308"    "SNP_mDIV_A4-4_081308"
 [7] "SNP_mDIV_A5-5_081308"    "SNP_mDIV_A6-6_081308"
 [9] "SNP_mDIV_A7-7_081308"    "SNP_mDIV_A8-8_081308"
[11] "SNP_mDIV_A9-57_082108"   "SNP_mDIV_A9-9_081308"
[13] "SNP_mDIV_B1-12_081308"   "SNP_mDIV_B2-13_081308"
[15] "SNP_mDIV_B3-14_081308"   "SNP_mDIV_B4-15_081308"
[17] "SNP_mDIV_B5-16_081308"   "SNP_mDIV_B6-17_081308"
[19] "SNP_mDIV_B7-18_081308"   "SNP_mDIV_D10-187_082108"
[21] "SNP_mDIV_D3-179_082108"

attr(,"class")
[1] "probesetSampleMatrixes" "list"
attr(,"matrixNames")
[1] "geno" "vino" "conf"
```

The function documentation for `genotypeSnps(...)` contains an explanation for all of the value codes shown.

You may be interested in visually inspecting the distribution of intensity contrasts and averages for a particular SNP across samples. If this is the case you can use the `plotSnpContAvg(...)` and `pointsSnpContAvg(...)` functions to do this. Here we will plot the first SNP (which is SNP number 15 in the original matrix):

```
> # all sample contrast averages for SNP15
> plotSnpContAvg(snpContAvg["SNP15", ])
> # now highlight the A alleles
> pointsSnpContAvg(
+     snpContAvg["SNP15", ],
+     renderMask=(genoVinoConf["SNP15", ]$geno == 1))
```

6 Further Discussion on Normalization and Genotyping Algorithms Used

This section provides more detailed information on the normalization methods and genotyping algorithms used in this package. This information is not essential to using the package but is very important in determining which parameters should be chosen and how to interpret results.

6.1 Normalization

MouseDivGeno offers three normalization steps: intensity bias correction due to C G content in probe sequences and restriction fragment length correction, quantile normalization based on a reference distribution, and median summarization. Each probe set of Mouse Diversity Array has different restriction fragment length and C G content, and it affects the intensity. To adjust those difference, we initially chose 350 .CEL files, fit a spline regression, and obtained the coefficients for each probe. These coefficitents were saved in the annotation files and is used to normalize a new array.

Quantile normalization is commonly used in microarray data to remove array specific noise, and the reference distribution is often derived from each batch of arrays. However obtaining a reference distribution each time introduces an unnecessary batch effect, so we derived one reference distribution using 350 training arrays, and saved this reference distribution. Note that the quantile normalization can only be applied to samples having the same underlying distribution to classical inbred strains (such as C57BL/6J). If there is sample whose intensity distribution is different from that of classical inbred strains then you should not use the quantile normalization option. `mouseDivDensityPlot(...)` can be a useful tool to check the intensity distribution.

After these normalization steps, we take a median from four probes intensities (two from sense and two from antisense strand) to make one probe set value. In some cases, probes on one strand perform much

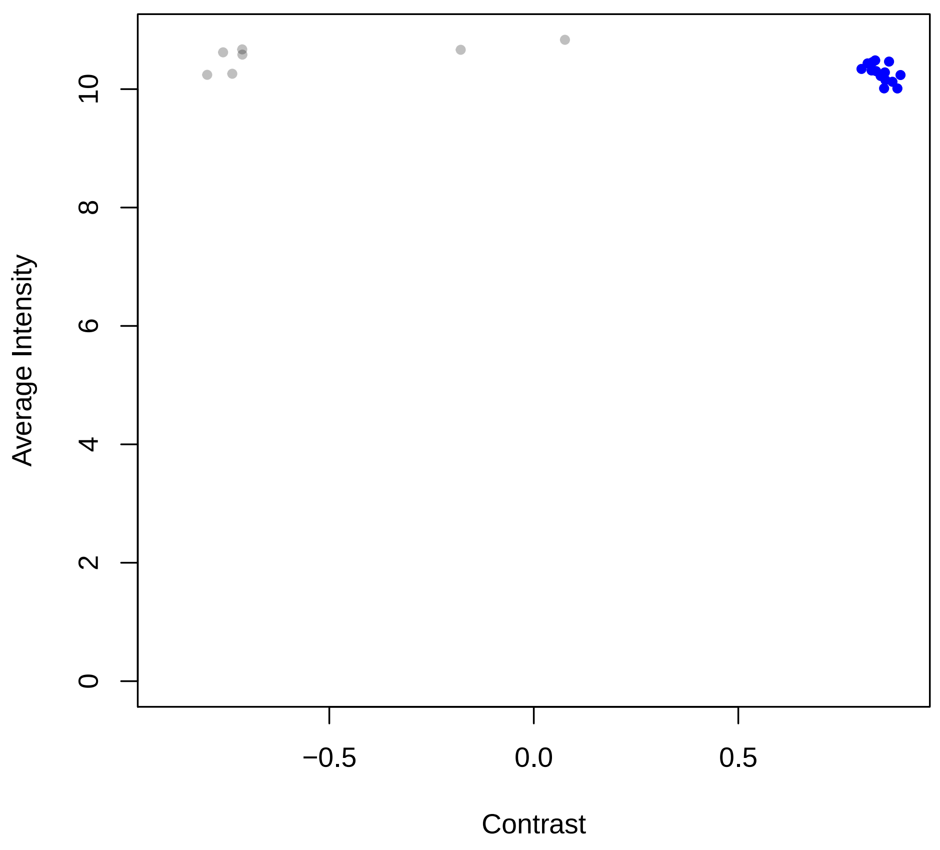

Figure 4: Distribution of Contrast and Average Intensities for SNP #15

worse than probes on the other strand. We compared intensities from sense and antisense, and removed 57,066 probes from sense strand and 61,196 probes from antisense strand. As a result for most probe sets we summarize four probe values, but for 118,262 probe sets we only summarize two probes.

6.2 Genotyping Algorithms

Genotyping is based on a method combining EM based clustering and single linkage hierarchical clustering. Detail algorithm follows. Suppose you genotype n samples.

1. test two groups (N = 2)

   (a) Find center via EM based clustering using contrast intensities.

      i. Initialization : μ_1 = maximum of contrast, μ_2 = minimum of contrast, $\sigma_1^2 = \sigma_2^2 = 0.1$

      ii. E step : calculate $P(j|i) = exp(-(x_i - \mu_j)^2/(2 * \sigma_j^2))$, where $j = 1, 2$ and $i = 1, \cdots, n$

      iii. M step : $\mu_j = \frac{\sum w_j y_i}{\sum w_j}$, $\sigma_j = \frac{\sum w_j * (y_i - \mu_j)^2}{\sum w_j}$, and $w_j = 1/N \sum p(j|i)$

   (b) Assign initial genotype. This step assigns genotype only for the samples having high probability $P(j|i)$. When the w_j is severely unbalanced the maximum $P(j|i)$ in a group could be quite small, and this initial genotype step tries to assign at least one sample to each genotype.

      i. For the group having a higher w_j (let's call this group j_1, and the other group j_2), let threshold = median of $P(j_1|i)$ only using sample i whose $P(j_2|i) < 0.5$. Assign genotype j_1 to samples i if $P(j_1|i)$ is bigger than the threshold.

      ii. Assign genotype for group j_2 : Threshold = find biggest mode of $P(j_2|i)$ only using unassigned i from the previous step. Also find median of $P(j_2|i)$ only using i whose $P(j_1|i) < 0.5$. Threshold is maximum of two values. Assign genotype j_2 to i if $P(j_2|i)$ is bigger than the threshold.

   (c) Genotype remaining samples using single linkage hierarchical clustering.

      i. Find one unassigned sample having the smallest distance to any assignment sample.

      ii. Assign the unassigned sample to the same genotype which the closest assigned sample belongs to.

      iii. Repeat this procedure till every sample gets genotyped.

2 test N = 3 : same as N = 2 except now the initialization : μ_1 = maximum of contrast, μ_2 = obtain from hint file or 0, and μ_3 = minimum of contrast.

3 Finalize the genotype. Compare N = 3 vs. N = 2 using silhouette score and distance between each group. If N = 3 fails, it compares N = 2 vs. N = 1.

VINOtyping Mouse Diversity Genotyping Array is based on C57BL6/J sequence and when a strain contains unknown SNPs (or a new mutation) somewhere in the probe sequence other than target SNP, hybridazation fails and it reduces the average intensity. Depending on the nature of the new mutation and the genotype of target SNP, identifying some VINOs is easier than identifying others. For instance when an inbred mouse has a new mutation right next to the target SNP, the hybridization failure is easy to detect. One the other hand, if the new mutation occurs at the end of the probe sequence, the hybridization failure may not be noticeable. When a strain has a AB genotype at the target SNP and has a new mutation, the intensity looks like homozygosity genotype with some reduction in intensity. Thus it is quite difficult to distinguish a homozygous genotype from heterozygosity with a new mutation in the probe sequence. Genotyping the parental strains helps to distinguish these cases.

Details on the approach we use for VINOtyping follows. To find VINO (variant intensity oligonecleotide), MouseDivGeno calculates the product of two probabilities. P(data is not a member of AA, AB, and BB) = 1 - P(data is a member of AA, AB or BB), and P(the intensity is low). P(data is a member of AA, AB or BB) is calculated by mohalanobis distance, and also returned as a confidence score. Note that when there are many VINOs they affect the mean and variance of each group, and to avoid this, we remove outliers and secondary cluster at the average intensity dimension if it exists. P(the intensity is low) is based on the average intensity. Again we removed the outliers and secondary cluster based on each genotype, then merge data from all genotypes to obtain the mean and variance of the average intensity, and calculate the probability based on the normal distribution. P(data is a VINO) is a product of those two probabilities, and using stringent threshold first it identifies samples having extremely low intensities. Then using single linkage based hierarchical clustering, it identifies samples clustered with the one having extremely low intensities, and finishes vinotyping. Note that when there is a VINO, the genotype of the original target SNPs should be considered as no call. It is because the observed intensities reflect the dynamic between a new mutation and the original genotype, and it is not obvious to predict the original SNP genotype. Also note that if VINOs are called in consecutive probes, it indicates a deletion. Thus vinotyping along with simple HMM can be used to detect deletion.

7 Generating LRR and BAF for PennCNV (Wang et al, 2007)

We will use the genotype data that we generated in the section titled ”Genotyping and Vinotyping for the Mouse Diversity Genotyping Array” along with the SNP and invariant data that we have already loaded in order to calculate LRR and BAF values. Before we start let's take a look at what is in the invariant data. One thing that you'll notice right away is that the invariant data types are lists of data frames rather than simple data frames. The reason for doing this is so that we could segregate the exons into two groups based on quality metrics and normalize them separately rather than normalizing them as a single group. More specifically the Mouse Diversity Genotyping Array contains 25mer sequences from exons which were carefully chosen for not containing any known SNP and thus can be used to study CNV. To get a better PCR amplification it was recommended that the restriction fragment length should be less than 1Kb, and when an exon probe set satisfies this condition we conventionally call this probe set exon1 and otherwise exon2 (Yang et al., 2008). This is why the invariant data structures are lists of length two.

```
> lapply(invariantProbeInfo, dim)

$exon1
[1] 747306      3

$exon2
[1] 448170      3

> lapply(invariantProbeInfo, function(x) x[1 : 2, ])

$exon1
  probeIndex   probesetId correction
1    2689437 Exon0000002a   1.114901
2    6819085 Exon0000002a   1.114901

$exon2
  probeIndex   probesetId correction
1    5940166 Exon0000001a   1.524253
2    4206500 Exon0000001a   1.524253

> lapply(invariantProbesetInfo, dim)

$exon1
[1] 373653      3

$exon2
[1] 224085      3

> lapply(invariantProbesetInfo, function(x) x[1 : 2, ])
```

```
$exon1
               probesetId chrId positionBp
Exon0000002a Exon0000002a     1    3206116
Exon0000002b Exon0000002b     1    3206555

$exon2
               probesetId chrId positionBp
Exon0000001a Exon0000001a     1    3092119
Exon0000001b Exon0000001b     1    3092179

> lapply(invariantReferenceDistribution, length)

$exon1
[1] 236896

$exon2
[1] 236896
```

See the documentation for `buildPennCNVInputFiles(...)` for a more detailed explaination of these input parameters.

In order to generate the LRR and BAF files we can enter a command like:

```
> dir.create(file.path(working_dir, "lrr-baf-output"))
> buildPennCNVInputFiles(
+     outdir                          = file.path(working_dir, "lrr-baf-output"),
+     allowOverwrite                  = TRUE,
+     genotypes                       = genoVinoResult$geno,
+     snpProbeInfo                    = snpProbeInfo,
+     snpInfo                         = snpInfo,
+     snpReferenceDistribution        = snpReferenceDistribution,
+     invariantProbeInfo              = invariantProbeInfo,
+     invariantProbesetInfo           = invariantProbesetInfo,
+     invariantReferenceDistribution  = invariantReferenceDistribution,
+     celFiles                        = file.path(working_dir, "celFiles"),
+     isMale                          = genoVinoResult$isMale,
+     chromosomes                     = c("19", "X", "Y", "M"))
```

When this command completes successfully the `lrr-baf-output` directory should contain the resulting LRR/BAF files along with a single `pfbdata.txt` file.

8 Simple CNV

The `simpleCNV(...)` function provides a simplified way to call CNVs which does not require using an external program such as PennCNV. Note that you must have the `HiddenMarkov` package installed in order to use this function. This function requires you to choose one CEL file to represent the ”reference” that CNV calls will be made against. All ”gains” and ”losses” are relative to this reference. Also note that there is an optional `summaryOutputFile` parameter. If you set this parameter to a file name then a summary report will be generated and written to that file, otherwise no summary report will be generated.

```
> library("MouseDivGeno")
> simpleCNVResult <- simpleCNV(
+     snpProbeInfo                    = snpProbeInfo,
+     snpInfo                         = snpInfo,
+     snpReferenceDistribution        = snpReferenceDistribution,
+     invariantProbeInfo              = invariantProbeInfo,
+     invariantProbesetInfo           = invariantProbesetInfo,
+     invariantReferenceDistribution  = invariantReferenceDistribution,
+     celFiles                        = file.path(working_dir, "celFiles"),
+     referenceCelFile                = file.path(working_dir, "celFiles/SNP_mDIV_A7-7_081308.CEL"),
+     chromosomes                     = c("19"),
+     summaryOutputFile               = file.path(working_dir, "vignetteCNVSummaryOut.txt"))
```

In the returned matrix list a value of 2 indicates no copy change, a value of 1 indicates a copy loss with respect to the reference and a value of 3 indicates a copy gain with respect to the reference. Now that we have calculated the CNVs for chromosome 19 we can ask questions like:

”Over all samples what percent of SNPs and invariants did we call as a CNV?”

```
> sum(simpleCNVResult$`19` != 2) / length(simpleCNVResult$`19`)
```

```
[1] 0.00145331
```

and, ”Which probesets did we call as CNVs for sample SNP_mDIV_B4-15_081308?”

```
> which(simpleCNVResult$`19`[, "SNP_mDIV_B4-15_081308"] != 2)
```

```
Exon0202262a Exon0202262b Exon0202262c  JAX00087083
       11027        11028        11029        11030
Exon0202263a Exon0202263b Exon0202263c Exon0202264a
       11031        11032        11033        11034
Exon0202264b Exon0202264c Exon0202265a Exon0202265b
       11035        11036        11037        11038
Exon0202265c Exon0202266a Exon0202266b Exon0202266c
       11039        11040        11041        11042
Exon0203882c Exon0203883a Exon0203883b Exon0203883c
       22790        22791        22792        22793
Exon0203884a Exon0203884b Exon0203884c
       22794        22795        22796
```

8.1 Further Details on CNV

simpleCNV(...) integrates normalized intensities from SNPs and exons. For SNP probe sets, the mean of average intensities of AB genotype group tends to be higher than that of AA or BB group, and to avoid intensity difference due to genotype group, simpleCNV(...) calculates intensities using max(A allele intensity, B allele intensity). We use HiddenMarkov an existing HMM R package to infer the most likely state from three possible states (1 = loss, 2 = normal, 3 = gain compared to the reference strain). Finally it saves the status and a summary table containing only copy number variance region. Compared to the first approach which obtains LRR and BAF from the canonical genotype grouping, this option is more useful to identify common CNV regions. On the other hand, this method relies on the reference strain, thus it is sensitive to the quality of the reference strain. For either approach users may ignore small size copy number variance region.

9 Acknowledgements

This work was funded by the NIGMS National Centers of Excellence in Systems Biology Program, grant P50 GM076468.

10 References

Wang, K.*et al* (2007) PennCNV: an integrated hidden Markov model designed for high-resolution copy number variation detection in whole-genome SNP genotyping data *Genome Research*, **17**, 1665-1674.

Yang, H.*et al* (2009) A customized and versatile high-density genotyping array for the mouse, *Nature Method*, **6**, 663-666.