

Advances in Microarray Probe Level Software

Benilton Carvalho Rafael Irizarry
carvalho@jhu.edu rafa@jhu.edu

Department of Biostatistics
Johns Hopkins University
Baltimore, MD

BioConductor Meeting, 2006

Motivation

Expression Arrays and the `affy` package

Preprocessing

The structure of `affy` package

Beyond Gene Expression Studies

SNP Arrays

Tiling Arrays

Infrastructure used with `oligo`

Examples

Genotype Calls with CRLMM

Creating PDEnvs

Handling Probe Intensities Data

Supplemental

Technology

Genotyping

Preprocessing

Algorithm

Acknowledgements

- ▶ James MacDonald, University of Michigan;
- ▶ Robert Scharpf, JHU;
- ▶ Terry Speed, UC Berkeley;
- ▶ Zhijin Wu, Brown University;

Expression microarrays

- ▶ Very popular tool in genetic research;
- ▶ Affymetrix GeneChip[®] is one of the most used commercial microarray;
- ▶ Manufacturers provide their own analytical software;
- ▶ Proprietary software often requires little user input.

Getting more from Expression GeneChip®

- ▶ Academic groups have developed methods that outperform manufacturer defaults;
- ▶ Availability of different options allows for flexible analyses;
- ▶ Multi-platform, open source and freely available solution further improves the scenario;
- ▶ The `affy` package is developed with this purpose in mind.

Preprocessing Microarray Data

- ▶ Image analysis;
- ▶ Background correction;
- ▶ Normalization;
- ▶ Summarization.

Background Correction

Probe intensities may be affected by:

- ▶ Non-specific binding;
- ▶ Optical noise.

The `affy` package offers alternative solutions:

- ▶ MAS 4.0 and 5.0;
- ▶ Signal and noise convolution (RMA).

Add-on packages offer other background correction methods:

- ▶ `gcrma` package;
- ▶ `vsn` package.

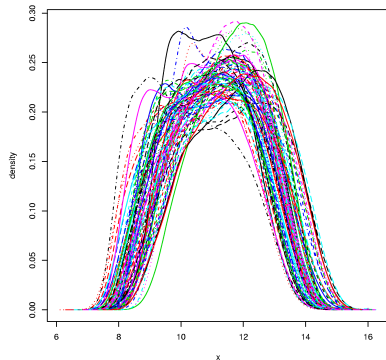
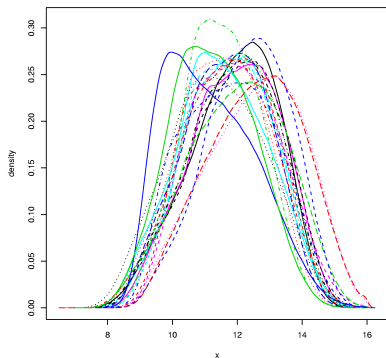
Normalization

- ▶ small changes in the quantity of RNA on different chips;
- ▶ chip to chip differences.

Normalization attempts to correct for systematic differences to make arrays comparable.

- ▶ loess;
- ▶ invariant set;
- ▶ quantiles.

Need for Normalization



Summarization

- ▶ Each genomic unit (gene, SNP, etc) is represented by multiple probes;
- ▶ Genomic quantity is thought as a summary of probe level quantities;
- ▶ Need to summarize probe level quantities within probe sets, for example:
 - ▶ MAS 5.0: Tukey's biweight;
 - ▶ RMA: median polish;
 - ▶ Li and Wong;
 - ▶ `PLIER` add-on.

Common concepts used in `affy`

- ▶ CDF environment;
- ▶ `AffyBatch`;
- ▶ MvA Plots;
- ▶ Robust Multi-Array Analysis;

`affy` is great, but what if we want to...

- ▶ make genotype calls?
- ▶ do copy number analysis?
- ▶ evaluate if transcription factors bind to particular regions on the chromatin?
- ▶ work with oligonucleotide arrays from other manufacturers?

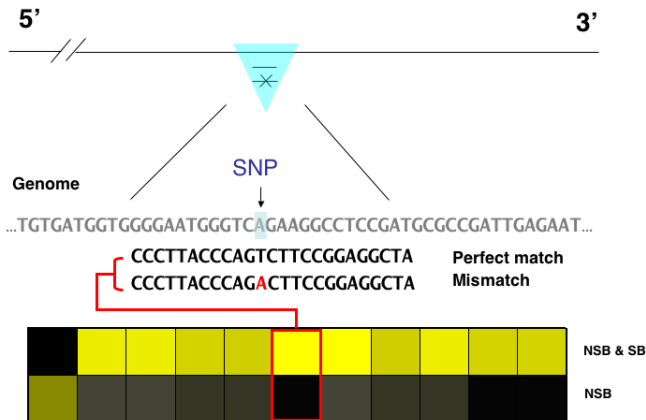
Different Arrays, but similar characteristics

Although there are different types of array, they share some commonalities:

- ▶ Multiple probes per genetic unit;
- ▶ Availability of sequence information;

These imply in similarities on some preprocessing aspects and methods already available can be used with slight modifications.

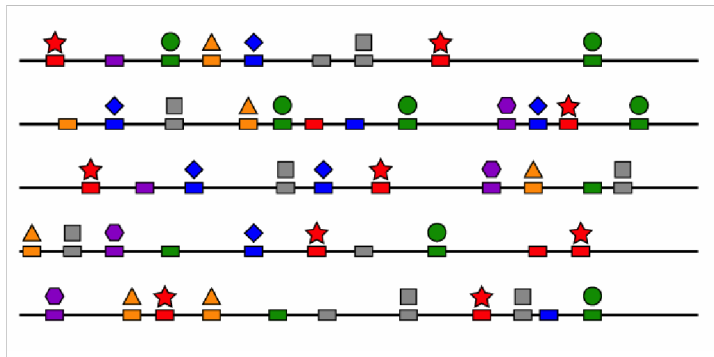
Design of the SNP Arrays



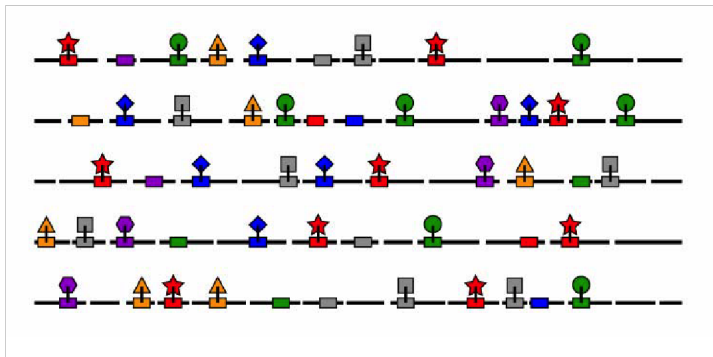
Design of the SNP Arrays



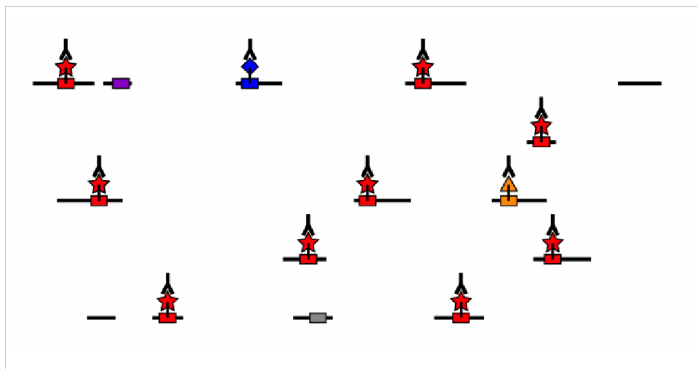
Design of the Tiling Arrays



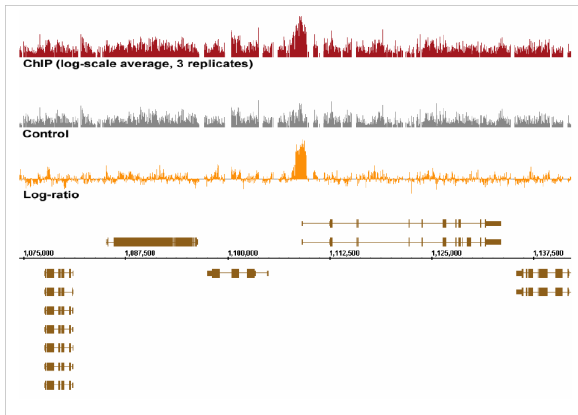
Design of the Tiling Arrays



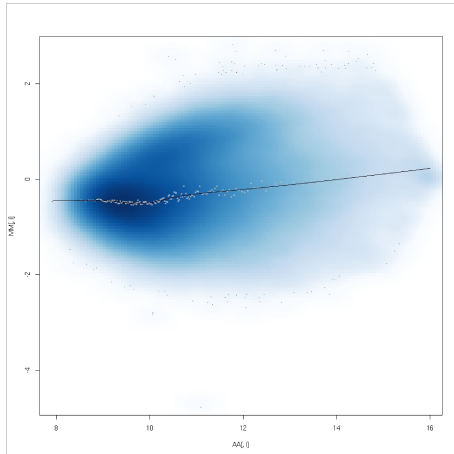
Design of the Tiling Arrays



Design of the Tiling Arrays



Design of the Tiling Arrays



How to handle different applications?

- ▶ Multiple tools for different applications;
- ▶ Different softwares for multiple manufacturers;
- ▶ Use the `oligo` package.

Platform Design Environment - PDEnv

- ▶ Information about the array design;
- ▶ Created by the `makePlatformDesign` package;
- ▶ Links probe intensities to the design;
- ▶ Contains: name, type, sequence, X/Y coordinates, allele, strand, snp position, genomic location.

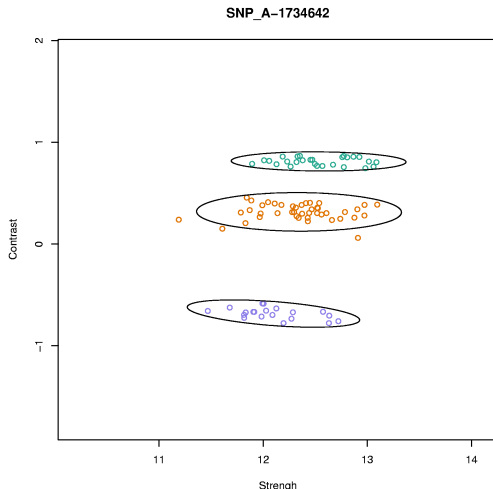
FeatureSet

- ▶ General container for probe intensities;
- ▶ Extends the eSet class;
- ▶ Has three subclasses:
 - ▶ Expression arrays - ExpressionFeatureSet objects;
 - ▶ SNP arrays - SnpFeatureSet objects;
 - ▶ Tiling arrays - TilingFeatureSet objects.

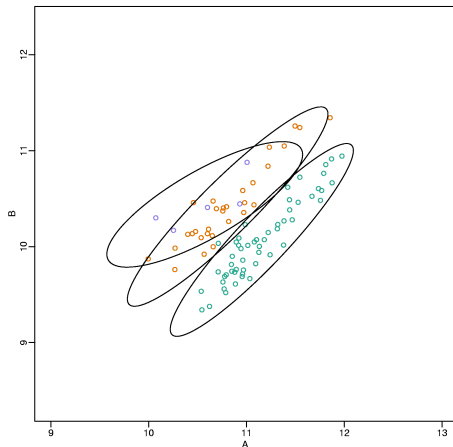
Containers for Summarized Data

- ▶ ExpressionSet - `rma()`;
- ▶ TilingSet;
- ▶ SnpQSet - `snprma()/justsnprma()`:
 - ▶ SenseThetaA;
 - ▶ SenseThetaB;
 - ▶ AntisenseThetaA;
 - ▶ AntisenseThetaB.

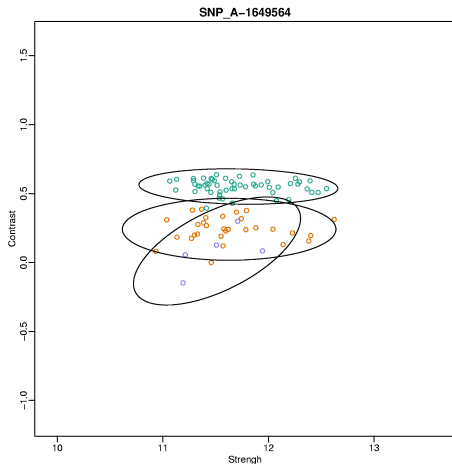
BRLMM Works in Some Cases



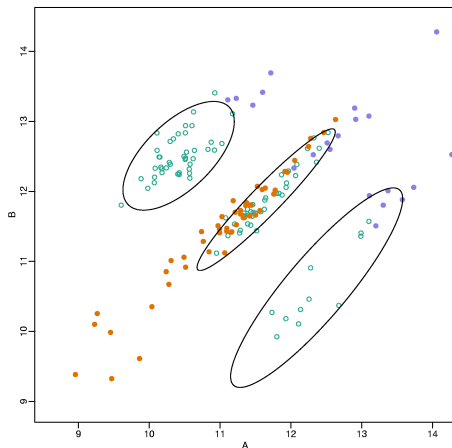
RLMM Fails Sometimes...



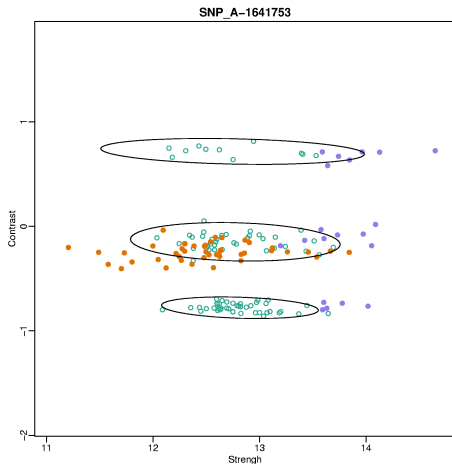
... and BRLMM Tries to Fix



RLMM Accross Labs



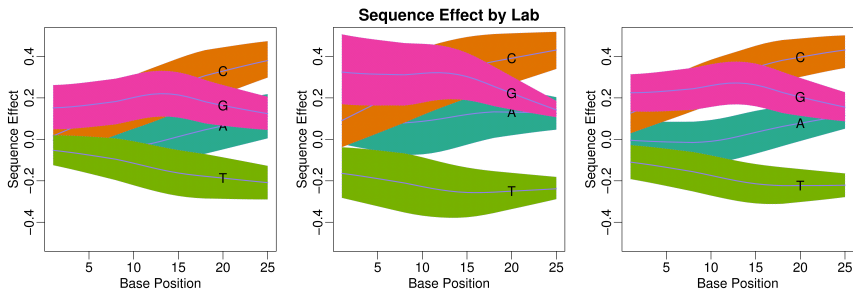
BRLMM Accross Labs



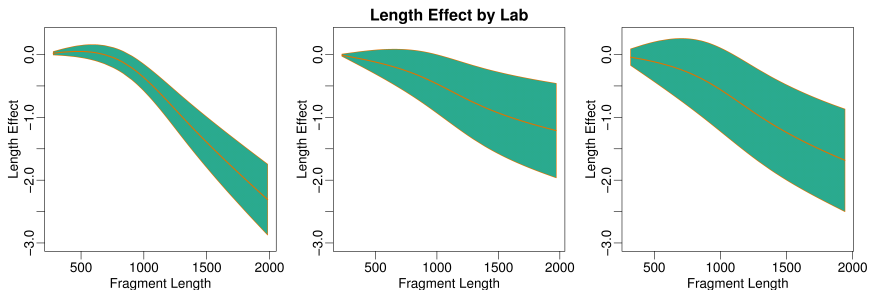
Different Sources of Variation on Observed Intensities

- ▶ Probe sequences;
- ▶ Fragment length;
- ▶ General factors change accross laboratories.

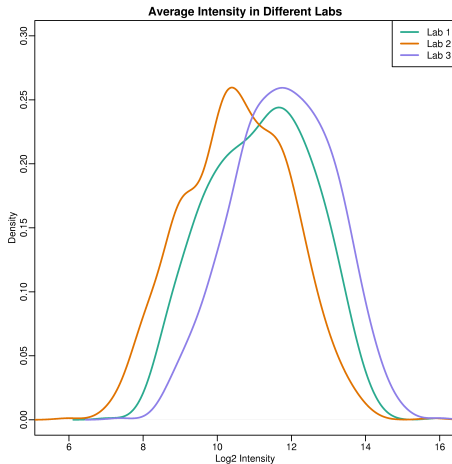
Different Sequences - Different Intensities



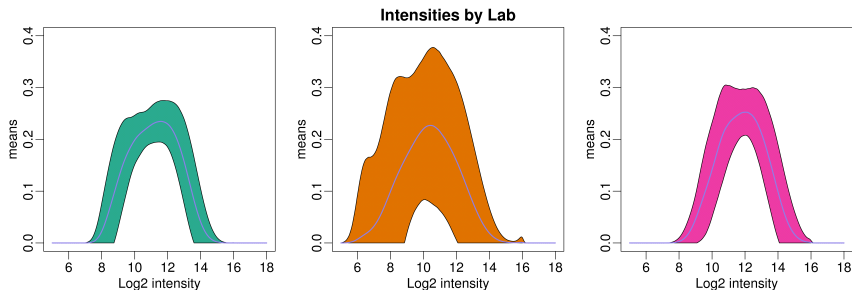
Different Lengths - Different Intensities



Different Labs - Different Intensities



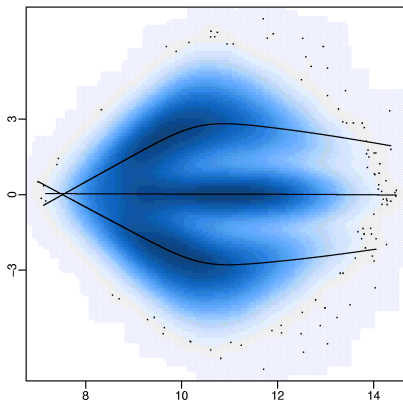
Same Lab - Different Intensities



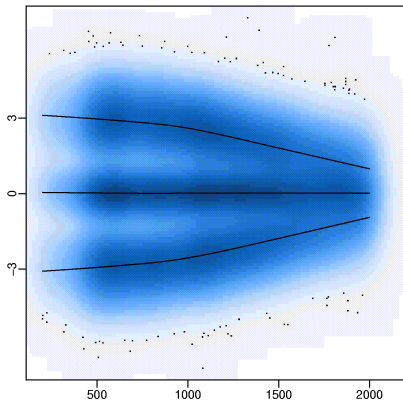
Summarization and Additional Corrections

- ▶ Allele specific intensities are summarized by median polish;
- ▶ Information on different strands are not combined;
- ▶ Fit a mixture model taking into account fragment length and average intensity on log ratios;

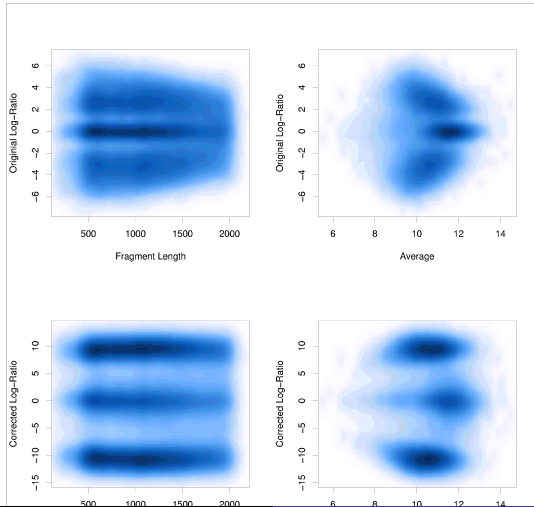
Log Ratio A/B vs. Average Intensity



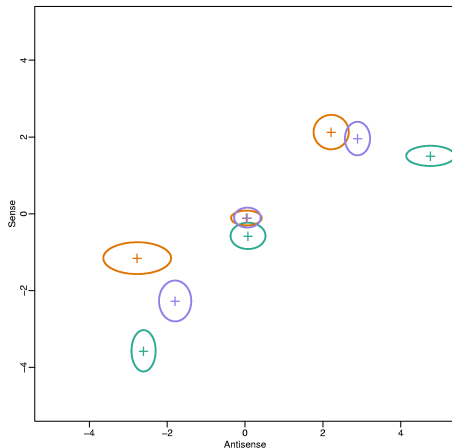
Log Ratio A/B vs. Fragment Length



Consequences of CRLMM



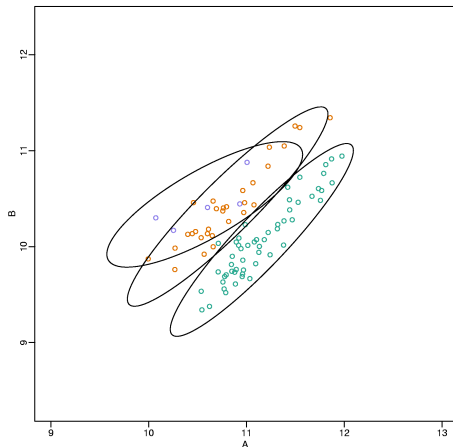
Different SNPs Have Different Parameters



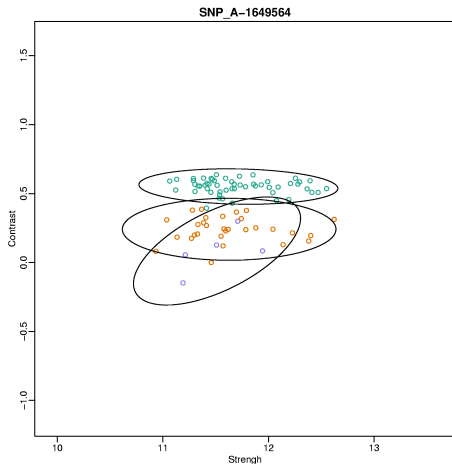
Model

$$\begin{aligned} [M_{i,j,s} | Z_{i,j} = k, m_{i,k}] &= f_{j,k}(X_{i,j,s}) + m_{i,k} + \epsilon_{i,j,k,s} \\ f_{j,2}(\cdot) &= 0 \\ f_{j,1}(\cdot) &= -f_{j,3}(\cdot) \\ \mathbf{m} &\sim N(0, \mathbf{V}) \\ \epsilon_{i,j,k,s} &\sim N(0, \sigma_{i,k,s}^2) \\ \sigma_{i,1,s} &= \sigma_{i,3,s} \\ \frac{1}{\sigma_{i,k}^2} &\propto \frac{1}{d_{0,k} s_{0,k}^2} \chi_{d_{0,k}}^2 \end{aligned}$$

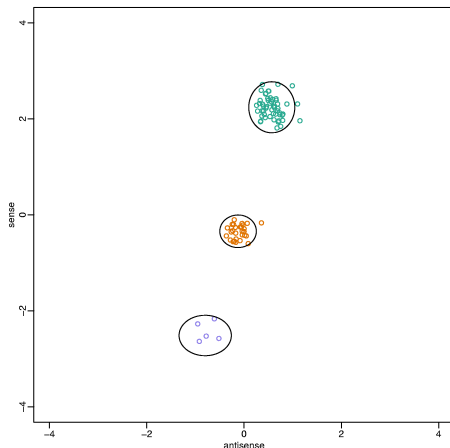
Two Strands Not Always Work: RLMM



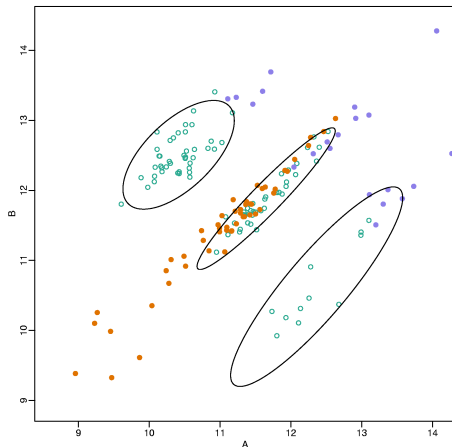
Two Strands Not Always Work: BRLMM



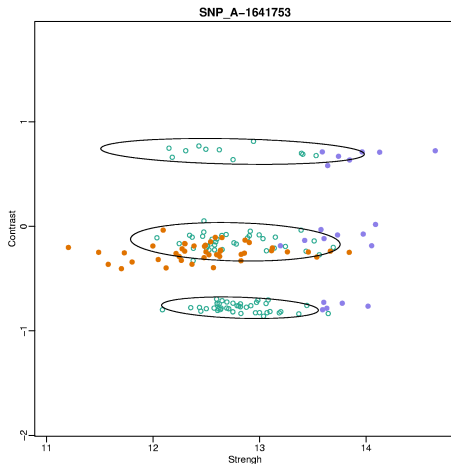
Broken Strands Do Not Break CRLMM



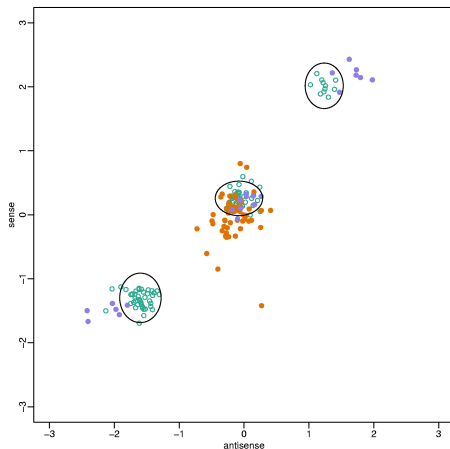
RLMM Accross Labs



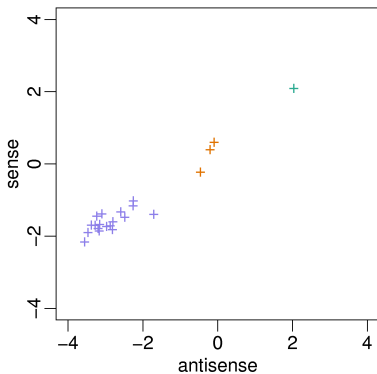
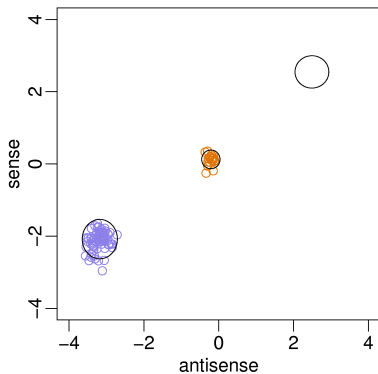
BRLMM Accross Labs



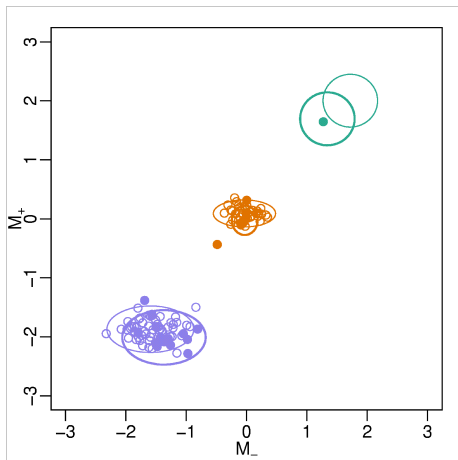
CRLMM Accross Labs



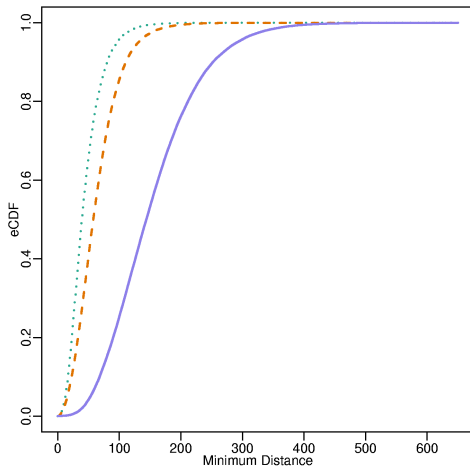
CRLMM and Missing Data



CRLMM Calibrates Means and Variances



C-RLMM Increases Separation



Running CRLMM on `oligo`

- ▶ Read data:
`read.celfiles()`
- ▶ Preprocess and summarize:
`snprma()`
- ▶ Run CRLMM on the `SnpQSet` object:
`crlmm()`

Remarks and Future Work

- ▶ C-RLMM is MM independent;
- ▶ The approach can be applied to technologies other than Affymetrix;
- ▶ The algorithm offers increased separation;
- ▶ Quality control on 100K arrays;
- ▶ Evaluation of CRLMM's performance on 500K arrays;
- ▶ Moving towards Copy Number Analysis.

Creating PDEnvs for Affymetrix Expression Arrays

The session below will create and install the PDEnv for the Mouse430_2 design.

```
> library(makePlatformDesign)
> makePDpackage("Mouse430_2.cdf",
                "Mouse430_2_probe_tab")
```

```
R CMD INSTALL pdmouse4302
```

Creating PDEnvs for Affymetrix SNP Arrays

```
> library(makePlatformDesign)
> makePDpackage("Mapping50K_Xba240.CDF",
                "Mapping50K_Xba240_probe_fasta",
                "Mapping50K_Xba240_annot.csv",
                type="SNP")
```

R CMD INSTALL pdmapping50kxba240

Creating PDEnvs for Affymetrix Tiling Arrays

```
> library(makePlatformDesign)
> makePDpackage("Chrom21-22B-2um.bpmap",
               type="tiling", genomebuild="hg15")
```

```
R CMD INSTALL pdchrom2122b2um
```

Creating PDEnvs for NimbleGen Expression Arrays

```
> library(makePlatformDesign)
> makePDpackage("2004-11-09_Human_60mer_TEST.ndf",
                "ngsExpression1.xys",
                manufacturer="nimblegen")
```

```
R CMD INSTALL pd20041109human60mertest
```

Creating PDEnvs for NimbleGen Tiling Arrays

```
> library(makePlatformDesign)
> makePDpackage("2005-03-10_HG17_promoter1.ndf",
               "ngsTiling1.XYS",
               "2005-03-10_HG17_promoter1.pos",
               manufacturer="nimblegen",
               type="tiling", genomebuild="hg17")
```

```
R CMD INSTALL pd20050310hg17promoter1
```


Affymetrix Expression Arrays

```
> library(oligo); library(OligoData); library(geneplotter)
> cpath = setwd()
> setwd(file.path(.path.package("OligoData"), "affySnp"))
> files = list.celfiles()
> affyExpression = read.celfiles(files)
> setwd(cpath)
```

Affymetrix Expression Arrays

```
> lPms = log2(pm(affyExpression)[,1])
> lMms = log2(mm(affyExpression)[,1])
> mBp = round(mean(lMms>lPms)*100, 2)
> title = paste("MM > PM in ", mBp, "% of the probes")
> smoothScatter(lMms, lPms, main=title,
                ylab="PM log intensity",
                xlab="MM log intensity",
                xlim=c(4,16), ylim=c(4,16))
> abline(coef=c(0,1))
```

Affymetrix Expression Arrays

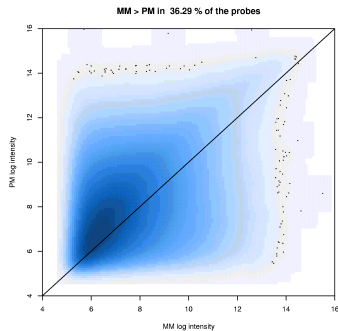


Figure: MM probes can be bigger than PM probes

Affymetrix SNP Arrays

```
> library(oligo); library(OligoData); library(genepLOTter)
> cpath = getwd()
> setwd(file.path(.path.package("OligoData"), "affySnp"))
> files = list.celfiles()
> xbaData = read.celfiles(files)
> setwd(cpath)
```

Affymetrix SNP Arrays

```
> xbaData = read.celfiles(files)
> pmi = pmindex(xbaData)
> pmAllelesAB = alleleAB(xbaData)[pmi]
> alleleA = pmAllelesAB == "A"
> lPms = log2(pm(xbaData)[,1])
> M = lPms[alleleA]-lPms[!alleleA]
> A = (lPms[alleleA]+lPms[!alleleA])/2
> smoothScatter(A, M, main="MvA Plot",
                 ylab="Log ratio (A/B)",
                 xlab="Average log intensity")
```

Affymetrix SNP Arrays

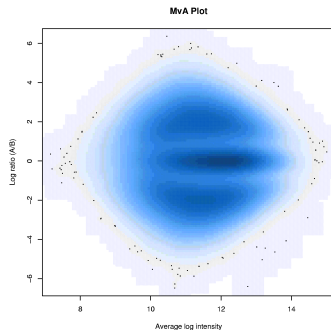


Figure: Log ratio (allele A vs. allele B) behaves differently depending on the genotype of the SNPs being considered

Plotting a region

```
> plotRegion = function(data, sample, lb, ub){  
  posit = pmPosition(data)  
  ok = !is.na(posit) & posit >= lb & posit < ub  
  x = posit[ok]  
  y = log2(pm(data)[ok, sample])  
  plot(x, y, pch=19, cex=.2, ylab="Log Intensity",  
        xlab="Genomic Position",  
        main="Log Intensities vs. Genomic Position")  
  lines(lowess(x, y, f=1/3), lwd=2, col="blue")  
}
```

Affymetrix Tiling Arrays

```
> library(oligo); library(OligoData); library(geneplotter)
> cpath = getwd()
> setwd(file.path(.path.package("OligoData"), "affyTiling"))
> files = list.celfiles()
> tilingData = read.celfiles(files)
> setwd(cpath)
> plotRegion(tilingData, 1, 4.392e7, 4.3924e7)
```


Affymetrix Tiling Arrays

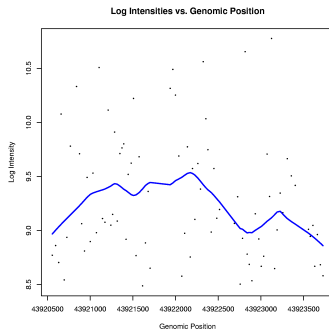


Figure: Log intensities observed between two genomic positions

NimbleGen Expression Arrays

```
> library(oligo)
> library(OligoData)
> cpath = getwd()
> setwd(file.path(.path.package("OligoData"), "ngsExpression"))
> files = list.files()
> ngsExpression = read.files(files)
> setwd(cpath)
> summarized = rma(ngsExpression)
> par(mfrow=c(2,1))
> hist(ngsExpression, which="pm", xlim=c(5,17))
> plotDensity(exprs(summarized), xlim=c(5,17))
```

NimbleGen Expression Arrays

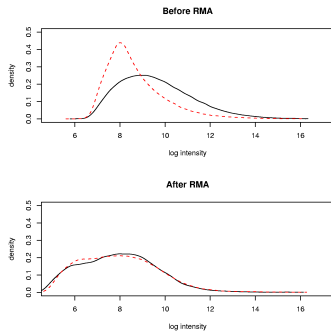


Figure: Effect of preprocessing using RMA

NimbleGen Tiling Arrays

```
> library(oligo)
> library(OligoData)
> cpath = getwd()
> setwd(file.path(.path.package("OligoData"), "ngsTiling"))
> files = list.xysfiles()
> ngsTiling = read.xysfiles(files)
> setwd(cpath)
> plotRegion(ngsTiling, 1, 4.392e7, 4.3924e7)
```

NimbleGen Tiling Arrays

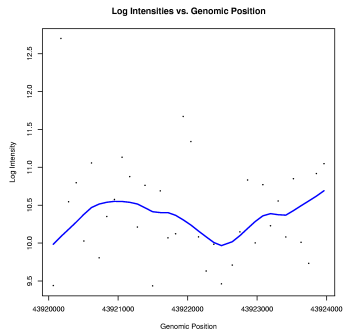






Figure: Log intensities observed between two genomic positions

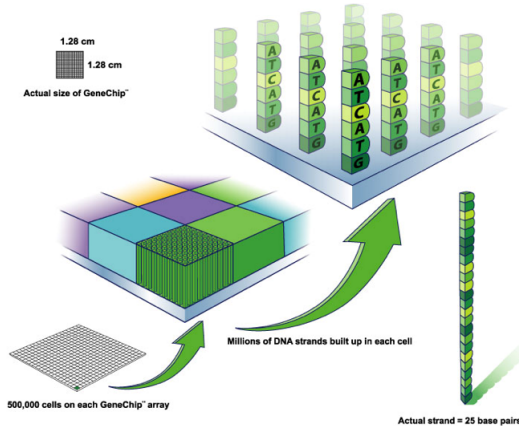
NimbleGen Tiling Arrays

```
> pmSeqs = pmSequence(ngsTiling)
> contents = basecontent(pmSeqs)
> contents[1:5,]
  A   T   C   G
21 13  14   9
22 19  10  14
22 13  12  11
25 17  12  12
20 17  11  12
```

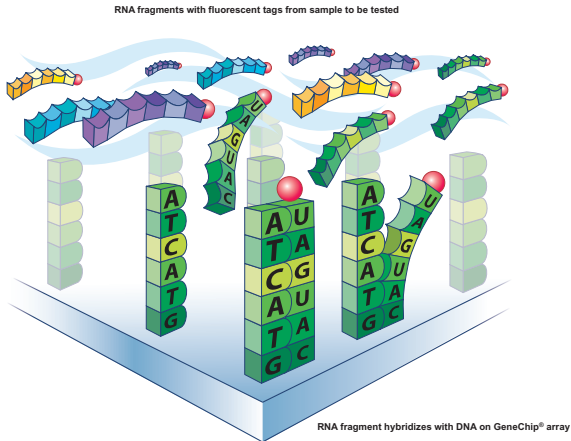
References

-  Di, X *et. al.*. *Bioinformatics* 2005;
-  Irizarry *et. al.*. *Biostatistics* 2003;
-  Li and Wong. *PNAS* 2001;
-  Rabbbe, N; Speed, TP. *Bioinformatics* 2006;

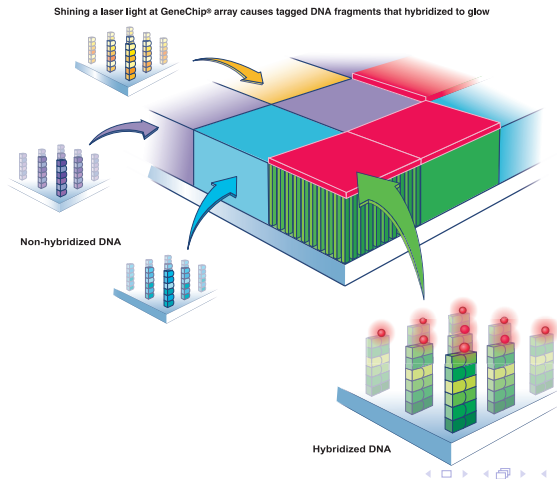
Design of the SNP Arrays



Hybridization



Hybridization



Genotype Calls: Why and How

Applications:

- ▶ Association studies;
- ▶ Linkage analyses;

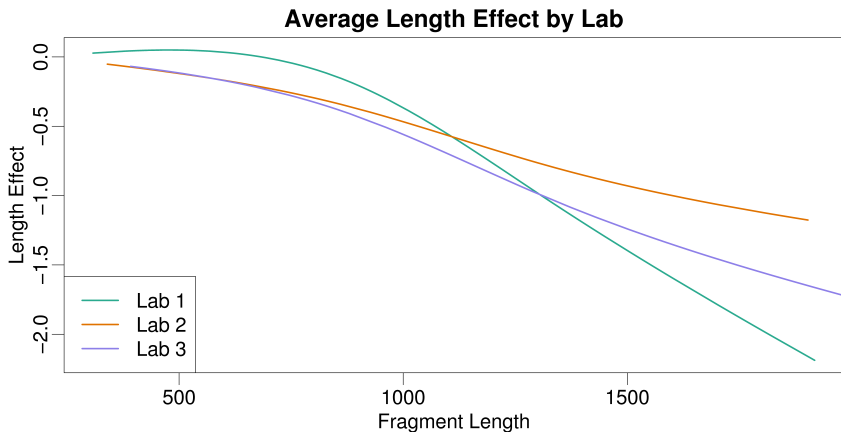
Available methods for genotyping:

- ▶ PCR;
- ▶ DNA Sequencing;
- ▶ Hybridization to DNA Arrays;

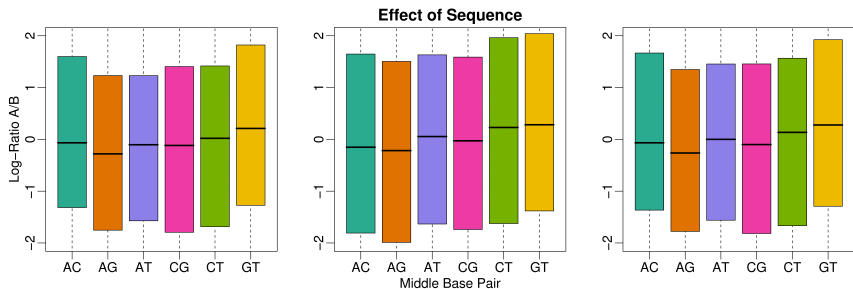
Some Algorithms for Genotype Calls

- ▶ PLASQ;
- ▶ DM;
- ▶ RLMM;
- ▶ B-RLMM;
- ▶ C-RLMM.

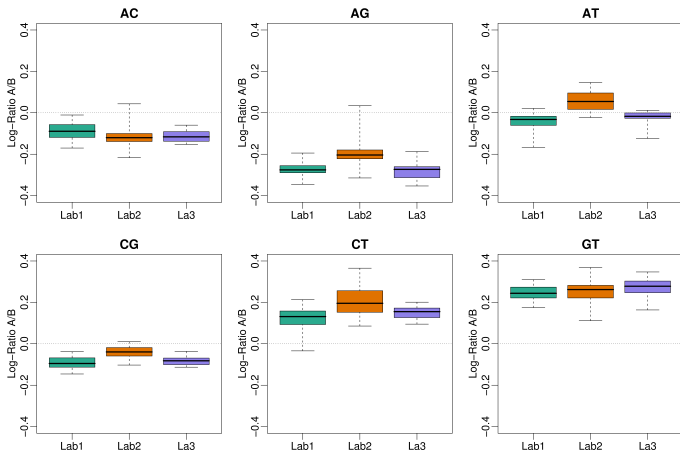
Different Lengths - Different Intensities



Log Ratio (A/B) and Middle Base Pair



(Median)Log Ratio (A/B) and Middle Base Pair



Algorithm - Preprocessing

- ▶ Read raw data;
- ▶ Adjust for fragment length and sequence;
- ▶ Quantile normalize;
- ▶ Apply median-polish on allele-specific intensities (+/-);
- ▶ Fit mixture model on log ratios;
- ▶ Estimated weights can be used to make initial calls;

Algorithm - Genotype Calls - Train on Hapmap Data

- ▶ Preprocess;
- ▶ For “well-defined” SNPs:
 - ▶ get μ, σ to form priors \mathbf{V}, s_0 and d_0 ;
 - ▶ Update parameters with shrinkage and keep μ, σ ;
- ▶ For remaining SNPs:
 - ▶ Use mixture to get initial calls;
 - ▶ Using initial calls, compute μ, σ ;
 - ▶ Update parameters;
- ▶ Keep all the updated parameters.

Algorithm - Genotype Calls - New Dataset

- ▶ Preprocess;
- ▶ Use μ, σ from Hapmap to get initial calls;
- ▶ Update using Bayes shrinkage;
- ▶ Get final calls using maximum likelihood.

Notation Used on Model

- ▶ $M_{i,j,s}$: log ratio (A/B) for SNP i on sample j ;
- ▶ Sense and antisense strands: $s = +, -$;
- ▶ $Z_{i,j}$: unknown genotype, $g = 1, 2, 3$ for AA, AB, and BB respectively.