GS01 0163
Analysis of Microarray Data

Keith Baggerly and Kevin Coombes
Department of Bioinformatics and Computational Biology
UT M. D. Anderson Cancer Center
kabagg@mdanderson.org
kcoombes@mdanderson.org

8 November 2007

Lecture 21: Comparing Microarray Analysis Methods

- Comparing ALL and MLL
- The ClassComparison Package
- Comparing processing methods
  - Shedden et al.
  - Wilcoxon rank-sum tests
  - Thresholds
  - Cope et al.

Comparing ALL and MLL

Early in the course (and in the latest homework), we looked at the ALL-MLL-AML data from the paper by Armstrong et al. in *Nature Genetics*, 2002; 30:41-47.

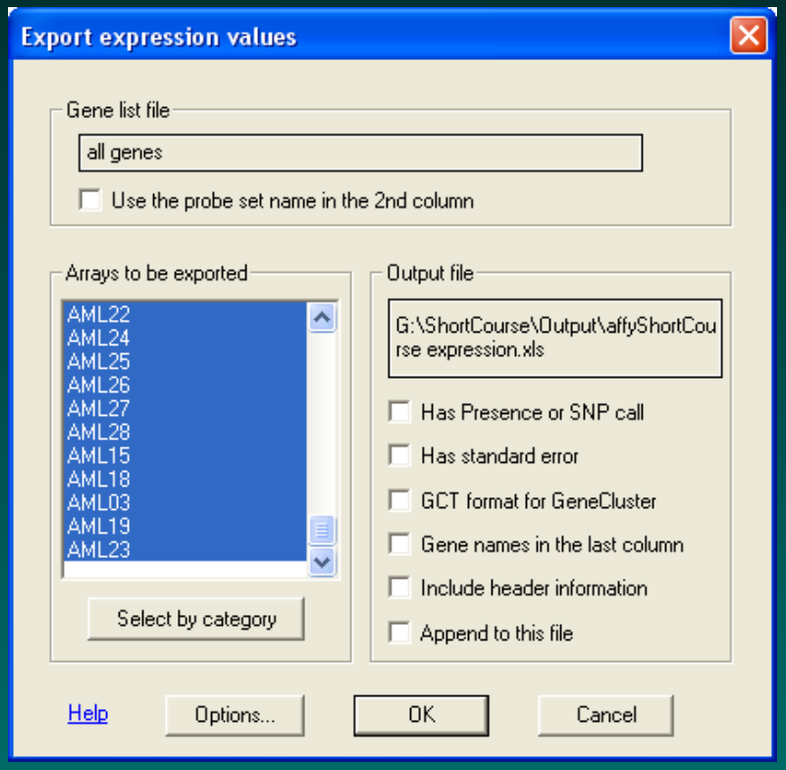
We learned that the AML samples were run on the U95Av2 chip, while the ALL and MLL samples were run on the U95A. We processed all the data in dChip (using a probe mask file to combine the two array types).

To analyze the data, we clustered the samples. We also found 628 differentially expressed genes between ALL and MLL.

We are now going to export the data from dChip and load it into R.

Exporting all the data from dChip

1. Use menu "Tools" $->$ "Export Expression Value".
2. Select "all genes"
3. Press "OK"



Starting in R

Load the `affy` package.

```
> require(affy)
Loading required package: affy
Loading required package: Biobase
Loading required package: tools
Welcome to Bioconductor
 Vignettes contain introductory material.  To view,
 simply type: openVignette()
 For details on reading vignettes, see
 the openVignette help page.
Loading required package: reposTools
[1] TRUE
```

Load the Sample Information file

```
> # remember where the data lives
> home <- 'g:/ShortCourse'
> # use the same sample info file we made for dChip
> si <- read.table(file.path(home, 'InfoFiles',
+                            'krc-sample-info.xls'),
+                  header=TRUE, sep='\t')
> si[1:5,]
          Scan.name type    Split
ALL01 CL20010111101AA  ALL Training
ALL02 CL20010111104AA  ALL Training
ALL03 CL20010111105AA  ALL Training
ALL04 CL20010111108AA  ALL Training
ALL05 CL20010111109AA  ALL Training
```

Load dChip's array file

```
> arrays <- read.table(file.path(home, 'Output',
                       'affyShortCourse arrays.xls'),
+                  header=TRUE, as.is=TRUE, sep='\t')
> # Fix the column names!
> dimnames(arrays)[[2]] <-
+     c(dimnames(arrays)[[2]][2:8], 'x')
> # Use the sample name as the row name
> dimnames(arrays)[[1]] <- arrays$Array
> # Only keep useful columns
> arrays <- arrays[, 3:6]
> # Give them sensible names
> dimnames(arrays)[[2]] <- c('MedianIntensity',
+     'PercentPresent', 'ArrayOutlier',
+     'SingleOutlier')
```

Combine sample information

```
> # Merge sample info with dChip info
> si <- merge(si, arrays, by='row.names',
+     sort=FALSE)
> # Sigh. Fix the row names yet again.
> dimnames(si)[[1]] <- si$Row.names
> # Remove redundant columns
> si <- si[, 2:8]
> rm(arrays) # cleanup
```

Note that the order has changed!

```
> si[1:5,]
           Scan.name type    Split MedianIntensity
ALL01 CL20010111101AA  ALL Training           1497
ALL24 CL20010111102AA  ALL     Test           1196
ALL02 CL20010111104AA  ALL Training           1778
ALL03 CL20010111105AA  ALL Training           1097
ALL04 CL20010111108AA  ALL Training           1489
      PercentPresent ArrayOutlier SingleOutlier
ALL01           48.2        1.648         0.099
ALL24           38.3        3.778         0.432
ALL02           49.5        1.450         0.144
ALL03           36.8        3.152         0.375
ALL04           38.7        5.299         0.216
```

Specialized factors

```
> # make a factor to compare ALL vs MLL
> temp <- si$type
> temp[temp=='AML'] <- NA
> si$ALLvMLL <- factor(temp)
>
> temp <- si$type
> temp[temp=='MLL'] <- NA
> si$ALLvAML <- factor(temp)
>
> temp <- si$type
> temp[temp=='ALL'] <- NA
> si$MLLvAML <- factor(temp)
```

```
> temp <- si$type
> temp[temp=='AML'] <- 'Other'
> temp[temp=='MLL'] <- 'Other'
> si$ALLvOther <- factor(temp)
>
> temp <- si$type
> temp[temp=='ALL'] <- 'Other'
> temp[temp=='MLL'] <- 'Other'
> si$AMLvOther <- factor(temp)
>
> temp <- si$type
> temp[temp=='AML'] <- 'Other'
> temp[temp=='ALL'] <- 'Other'
> si$MLLvOther <- factor(temp)
>
> si$type <- factor(si$type)
```

```
> summary(si)
  Scan.name            type        Split
 Length:72          ALL:24    Length:72
 Class :character   AML:28    Class :character
 Mode  :character   MLL:20    Mode  :character

MedianIntensity PercentPresent   ArrayOutlier
Min.   : 804    Min.   :28.30   Min.   : 0.253
1st Qu.:1222    1st Qu.:36.80   1st Qu.: 0.729
Median :1442    Median :41.10   Median : 1.085
Mean   :1483    Mean   :40.46   Mean   : 1.724
3rd Qu.:1727    3rd Qu.:44.83   3rd Qu.: 1.697
Max.   :3097    Max.   :49.80   Max.   :14.337
```

```
SingleOutlier     ALLvMLL    ALLvAML    MLLvAML
Min.   :0.0440    ALL :24    ALL :24    AML :28
1st Qu.:0.1610    MLL :20    AML :28    MLL :20
Median :0.2405    NA's:28    NA's:20    NA's:24
Mean   :0.2741
3rd Qu.:0.3460
Max.   :0.9520

ALLvOther   AMLvOther   MLLvOther
ALL  :24    AML  :28    MLL  :20
Other:48    Other:44    Other:52
```

Create the phenoData object

```
> pd <- new('phenoData', pData=si, varLabels=list(
+     Scan.name='CEL file name',
+     type='Histological classification',
+     Split='Used as training or test,
+     MedianIntensity='Unnormalized median brighness,
+     PercentPresent='Percentage of present calls',
+     ArrayOutlier='Percentage of Array Outliers',
+     SingleOutlier='Percentage of Single Outliers',
+     ALLvMLL='binary classifier',
+     ALLvAML='binary classifier',
+     MLLvAML='binary classifier',
+     ALLvOtherL='binary classifier',
+     AMLvOther='binary classifier',
+     MLLvOther='binary classifier'))
```

Create the MIAME object

MIAME = minimum information about a microarray experiment

Some of the BioConductor routines require a MIAME object, even though they will let you submit a character string as a description.

```
> miame <- new('MIAME',
+              name='SA Armstrong',
+              lab='Lander-Golub',
+              title='MLL translocations')
```

Read in the data from dChip

```
> temp <- read.table(file.path(home, 'Output',
+           'affyShortCourse expression.xls'),
+         header=TRUE, as.is=TRUE, sep='\t',
+         quote='', comment.char='')
> # expression data in the later columns
> data <- as.matrix(temp[, 6:77])
> # gene identifiers in the first five columns
> gi <- temp[, 1:5]
> # Use probe sets as row names
> dimnames(gi)[[1]]   <- gi$probe.set
> dimnames(data)[[1]] <- gi$probe.set
```

Check that the order agrees

We noticed that the order of entries in the sample info file had changed when we merged it with the dChip array information. Just to be on the safe side, we should make sure that the order of the data columns matches the sample infor rows.

```
> sum(dimnames(si)[[1]] != dimnames(data)[[2]])
[1] 0
> sum(dimnames(si)[[1]] == dimnames(data)[[2]])
[1] 72
```

Turn the dChip data into an exprSet

We can bring the dChip quantifications directly into R and turn them into an `exprSet`. Note that this avoids the memory problems by not brininging in the individual CEL files and not producing an AffyBatch.

```
> dchip <- new('exprSet',
+              exprs=data,
+              phenoData=pd,
+              annotation='hgu95av2',
+              description=miame,
+              notes='processed by KRC in dChip')
> rm(temp, data, si) # cleanup
```

justRMA

Meanwhile, we also loaded the CEL files and processed the data usaing `just.rma`. Since you will have done that in the homework, I am not going to put the code to do that here. In our analysis, the `exprSet` created by `just.rma` was named `rmaData`.

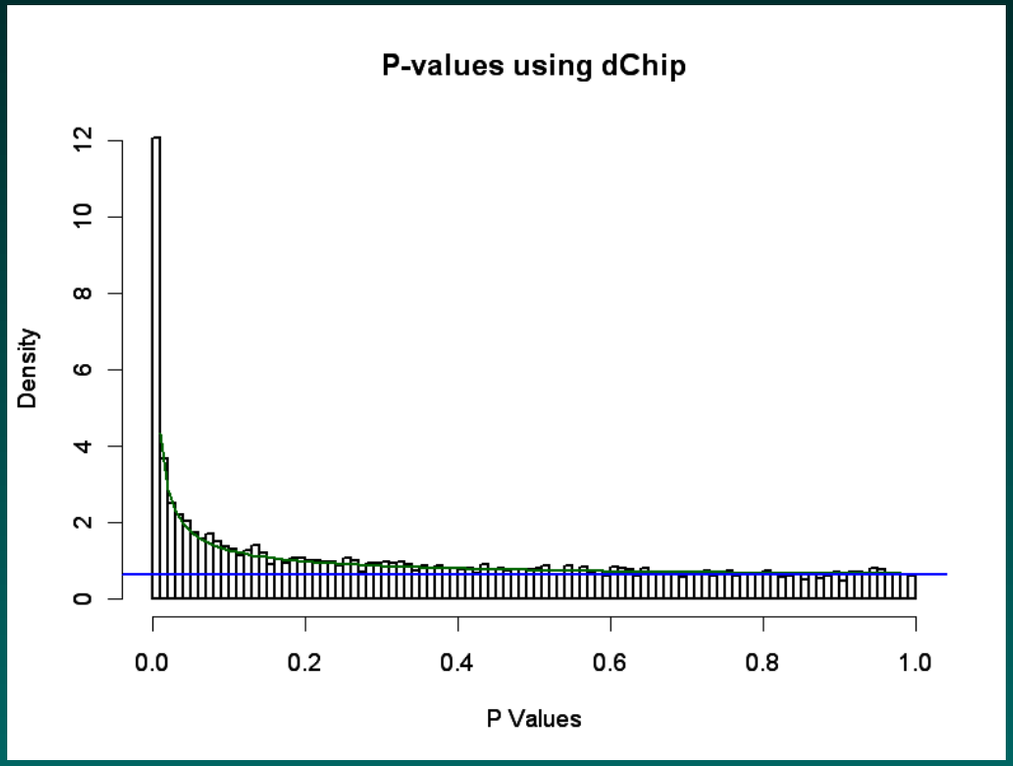
We are now going to start comparing the results of these two processing methods with respect to finding differentially expressed genes.

The ClassComparison Package

```
> require(ClassComparison)
Loading required package: ClassComparison
Loading required package: splines
Loading required package: oompaBase
Loading required package: PreProcess
Creating a new generic function for 'plot' in
 'PreProcess'
Creating a new generic function for 'print' in
 'PreProcess'
Creating a new generic function for 'as.data.frame'
 in 'PreProcess'
[1] TRUE
```

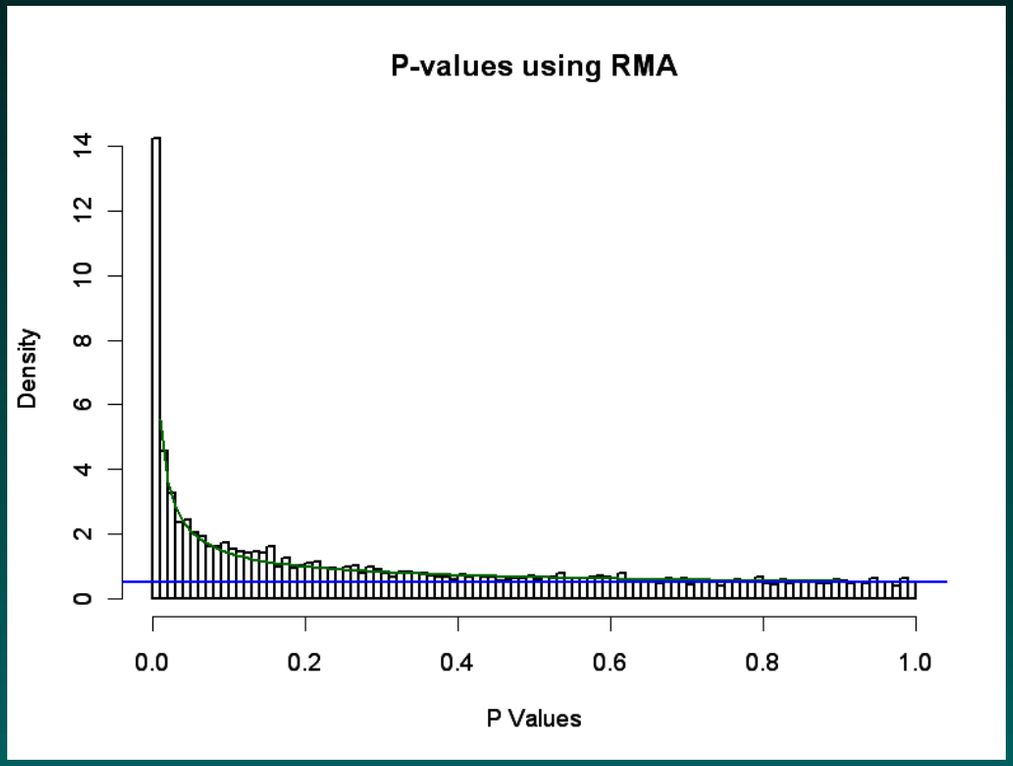
T-test, take one

```
> notAML <- pd@pData$type != 'AML'
> dchip.t <- MultiTtest(dchip[, notAML], 'ALLvMLL')
> dchip.b <- Bum(dchip.t@p.values)
> hist(dchip.b, main='P-values using dChip')
```



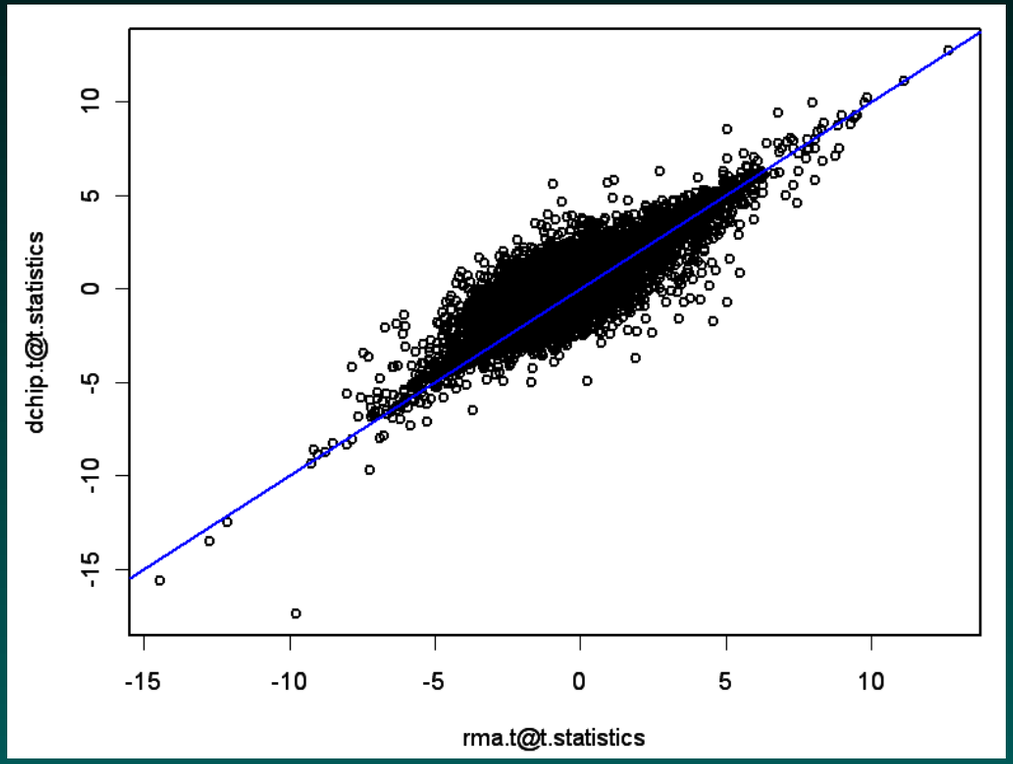
T-test, take two

```
> rma.t <- MultiTtest(rmaData[, notAML], 'ALLvMLL')
> rma.b <- Bum(rma.t@p.values)
> hist(rma.b, main='P-values using RMA')
```



Do the two methods agree?

```
> plot(rma.t@t.statistics, dchip.t@t.statistics)
> abline(0,1, col='blue')
```



How do we tell if the methods agree?

We have seen that the BUM plots for t-tests when we used different quantifications methods look similar. We have also seen that the t-statistics roughly agree, in the sense that they more or less follow the identity line. The haze around that line is rather "fat", however, which suggests that the exact lists of genes we get with the two methods may not be quite the same. Here's one difference:

```
> alpha <- 0.05
> countSignificant(dchip.b, alpha=alpha, by='FDR')
[1] 1520
> countSignificant(rma.b, alpha=alpha, by='FDR')
[1] 2353
```

Try a smaller FDR

```
> alpha <- 0.01
> countSignificant(dchip.b, alpha=alpha, by='FDR')
[1] 681
> countSignificant(rma.b, alpha=alpha, by='FDR')
[1] 992
```

There certainly appear to be a lot of differentially expressed genes. However, RMA seems to give us more genes than dChip at the same level of the False Discovery Rate. That already tells us something about the processing methods.

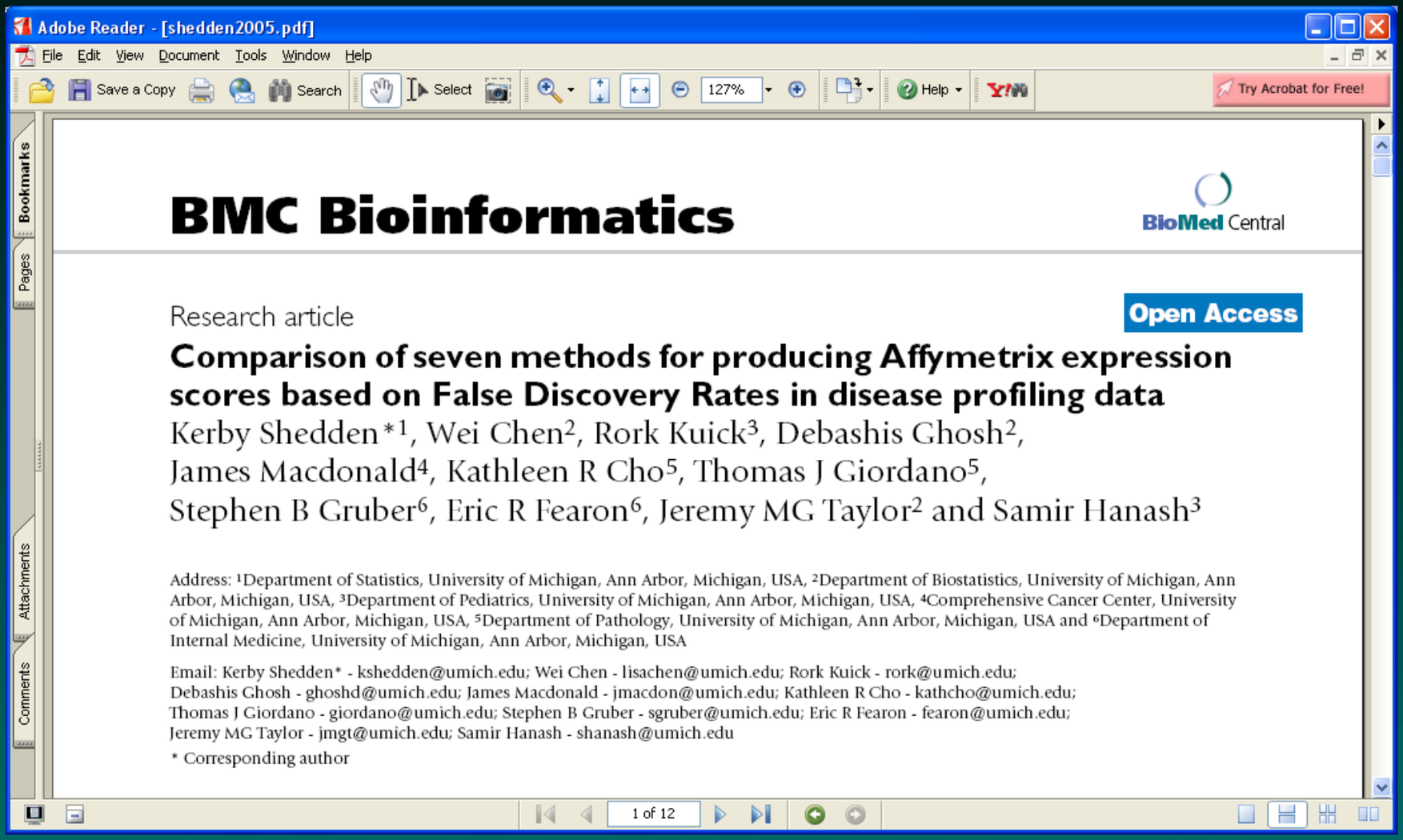
How much do the lists overlap?

```
> # logical vector: what does dChip find?
> dchip.01 <- selectSignificant(dchip.b,
+     alpha=alpha, by='FDR')
> # logical vector: what does RMA find?
> rma.01 <- selectSignificant(rma.b,
+     alpha=alpha, by='FDR')
> # Count the overlap
> sum(dchip.01 & rma.01)
[1] 563
> 563/681
[1] 0.8267254
```

Only 83% of the 681 genes found by dChip are contained in the larger list of genes found by RMA.

Comparing processing methods

So, the answers are “different”. Can we tell which is “better”?



Shedden et al., BMC Bioinformatics 2005; 6:26

They looked at 7 processing methods in two different data sets.

- Methods
  - dChip
  - GCRMA-EB
  - GCRMA-MLE
  - MAS5
  - PDNN
  - RMA
  - trimmed mean (TM)
- Data Sets
  - 47 Colon cancer, U133A (40 MSS vs. 7 MSI)
  - 79 Ovarian cancer, U133A (38 endometroid vs. 41 serous)

Shedden Fig1: Number of probe sets by FDR

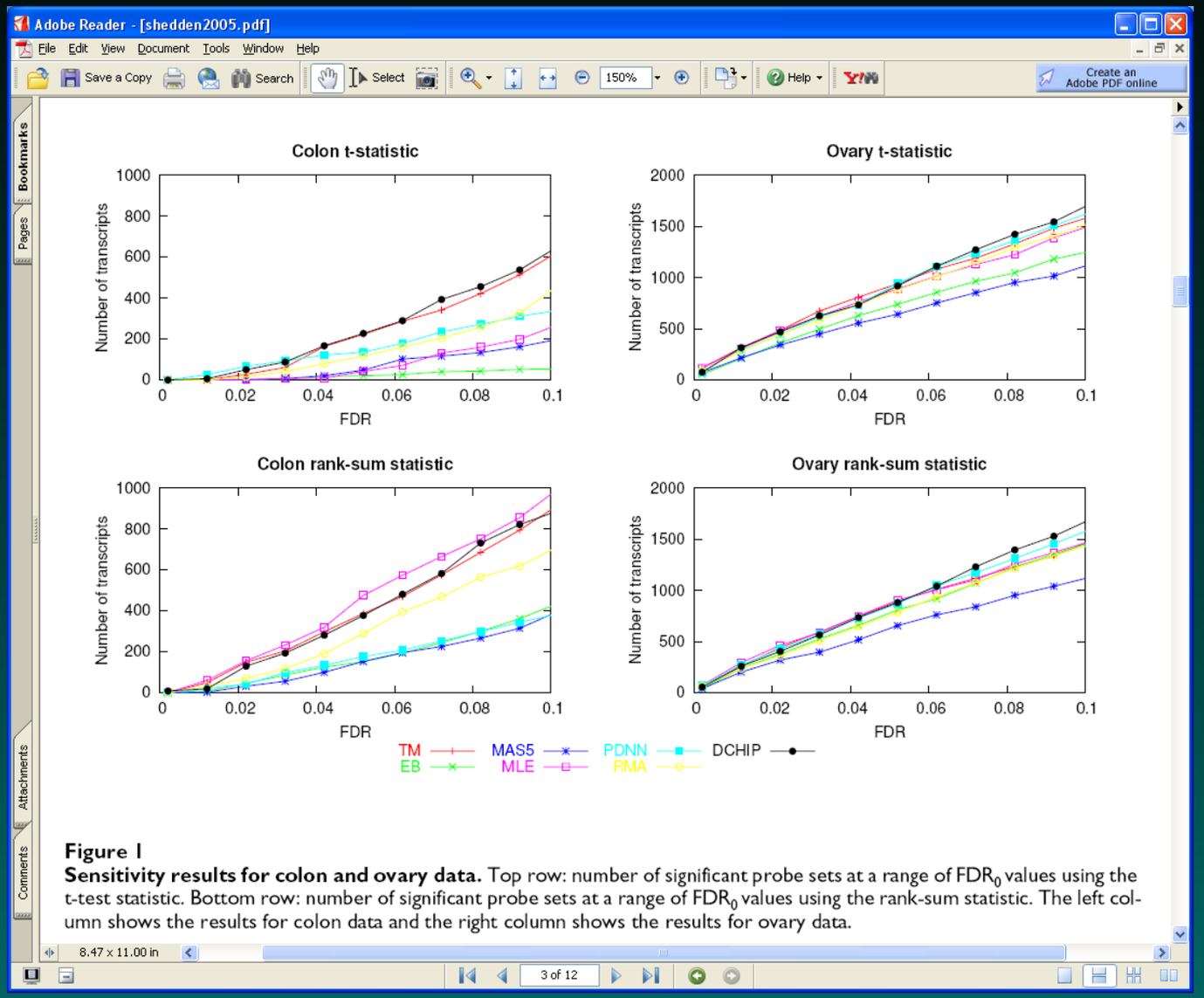


Figure 1
Sensitivity results for colon and ovary data. Top row: number of significant probe sets at a range of FDR_0 values using the t-test statistic. Bottom row: number of significant probe sets at a range of FDR_0 values using the rank-sum statistic. The left column shows the results for colon data and the right column shows the results for ovary data.

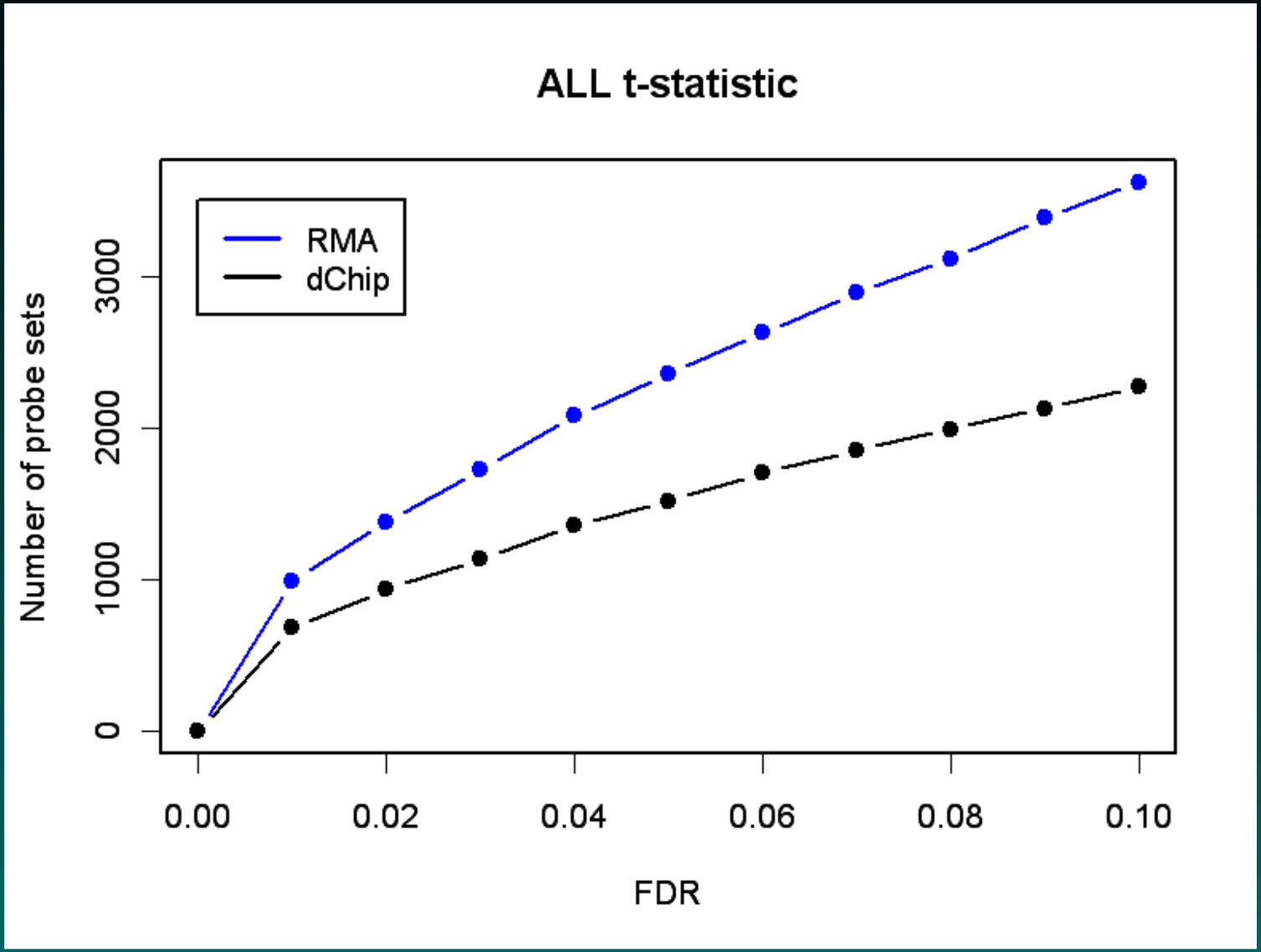
Same idea, ALL-MLL data set

```
> alpha <- seq(0, 0.1, by=0.01)
> f <- function(a, data) {
+   countSignificant(data, alpha=a, by='FDR')
+ }
>
> dchip.counts <- sapply(alpha, f, dchip.b)
> dchip.counts
 [1]    0  681  936 1139 1356 1520 1703
 [8] 1850 1990 2126 2266
> rma.counts   <- sapply(alpha, f, rma.b)
> rma.counts
 [1]    0  992 1379 1725 2078 2353 2623
 [8] 2890 3112 3383 3618
```

Making the plot

```
> plot(alpha, rma.counts,
+      xlab='FDR', ylab='Number of probe sets',
+      main='ALL t-statistic', type='b',
+      pch=16, col='blue')
> lines(alpha, dchip.counts, type='b', pch=16)
> legend(0, 3500, c('RMA', 'dChip'), lwd=3,
+      col=c('blue', 'black'))
```

RMA gives more differences in this data set



Wilcoxon rank-sum tests

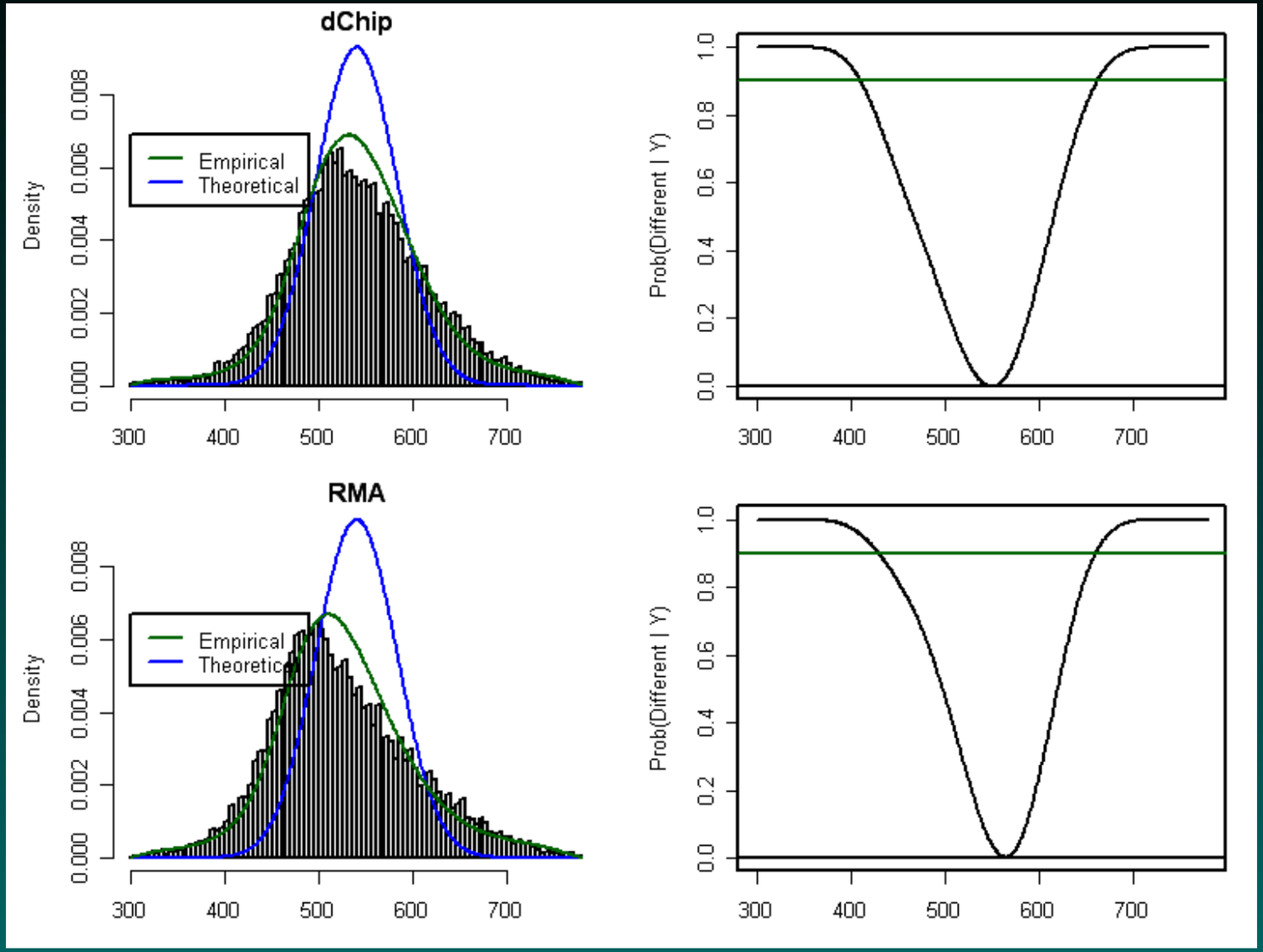
In the Shedden paper, they use a rank-sum statistic to test for differential expression, in addition to the t-statistic. To compute these statistics, we use the `MultiWilcoxonTest` function in the `ClassComparison` package.

```
> dchip.wil <- MultiWilcoxonTest(dchip[, notAML],
+     'ALLvMLL')
> rma.wil <- MultiWilcoxonTest(rmaData[, notAML],
+     'ALLvMLL')
```

Summary plots from the Wilcoxon empirical Bayes

```
> opar <- par(mai=c(0.5, 0.7, 0.2, 0.2),
+       mfrow=c(2,2))
> hist(dchip.wil, main='dChip')
> plot(dchip.wil, prior=0.725,ylim=c(0,1))
> abline(h=0)
> hist(rma.wil, main='RMA')
> plot(rma.wil, prior=0.56,ylim=c(0,1))
> abline(h=0)
> par(opar)
```

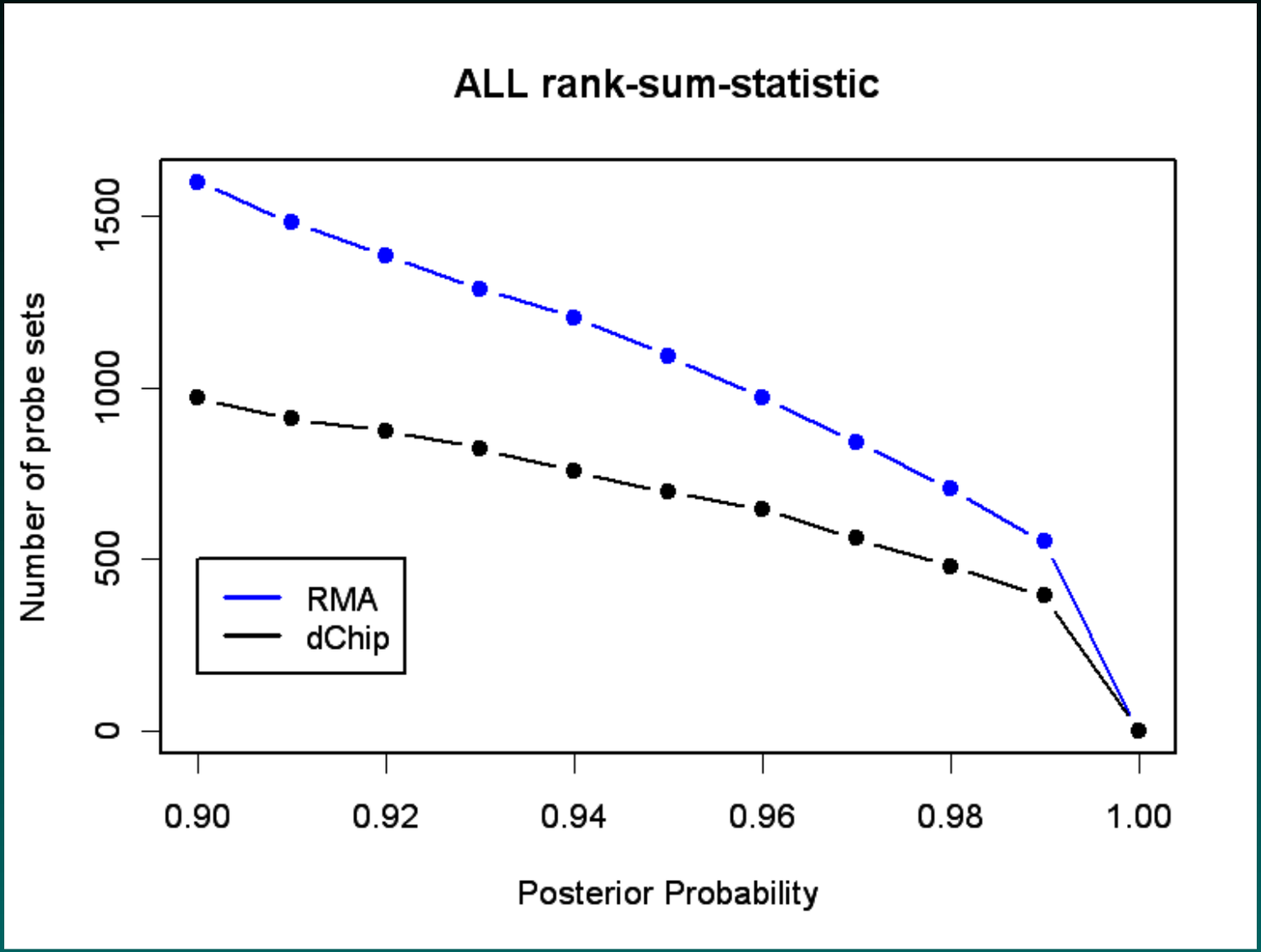
Why is the RMA version skewed?



Counts as a function of posterior probability

```
> sig <- seq(1.0, 0.9, by=-0.01)
> f2 <- function(s, p, data) {
+   countSignificant(data, prior=p, signif=s)
+ }
> dchip.w.counts <- sapply(sig, f2, p =0.725,
+     data=dchip.wil)
> dchip.w.counts
 [1]    0  394  481  562  648  695  756  824
 [9]  874  908  971
> rma.w.counts <- sapply(sig, f2, p =0.56,
+     data=rma.wil)
> rma.w.counts
 [1]    0  551  707  842  971 1091 1204 1288
 [9] 1384 1479 1598
```

RMA still gives more differences



Shedden Fig3: Threshold Statistic by FDR

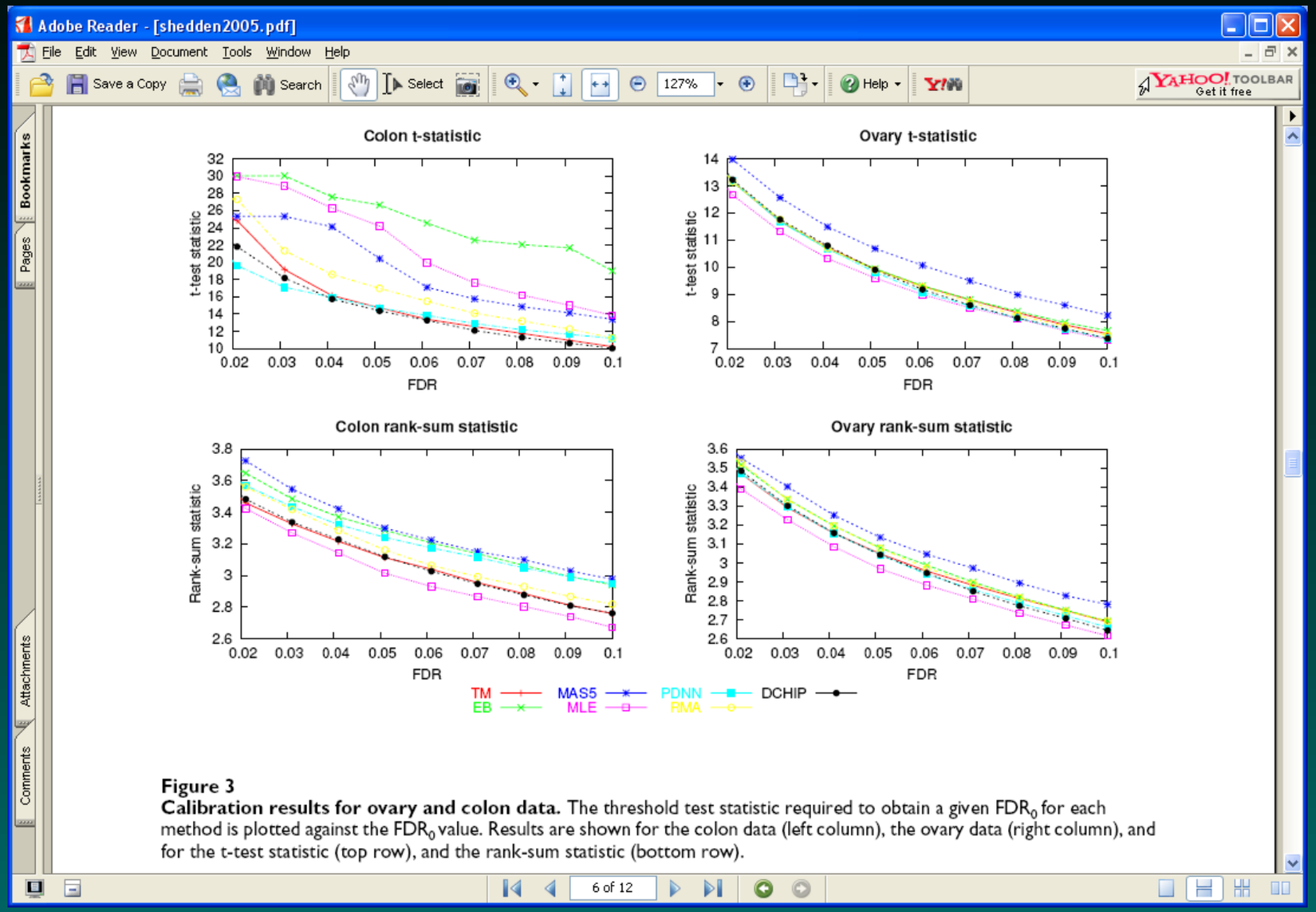
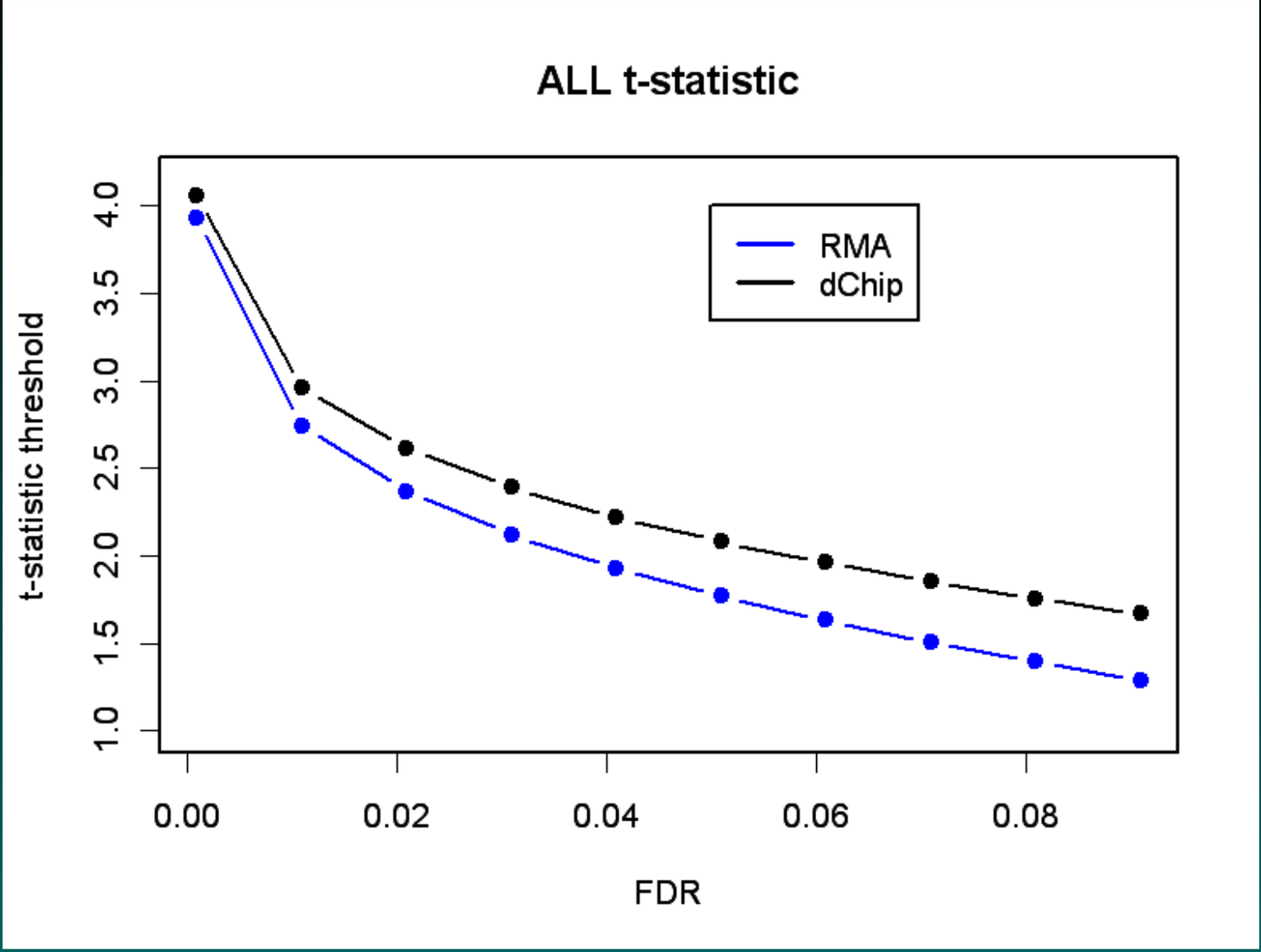


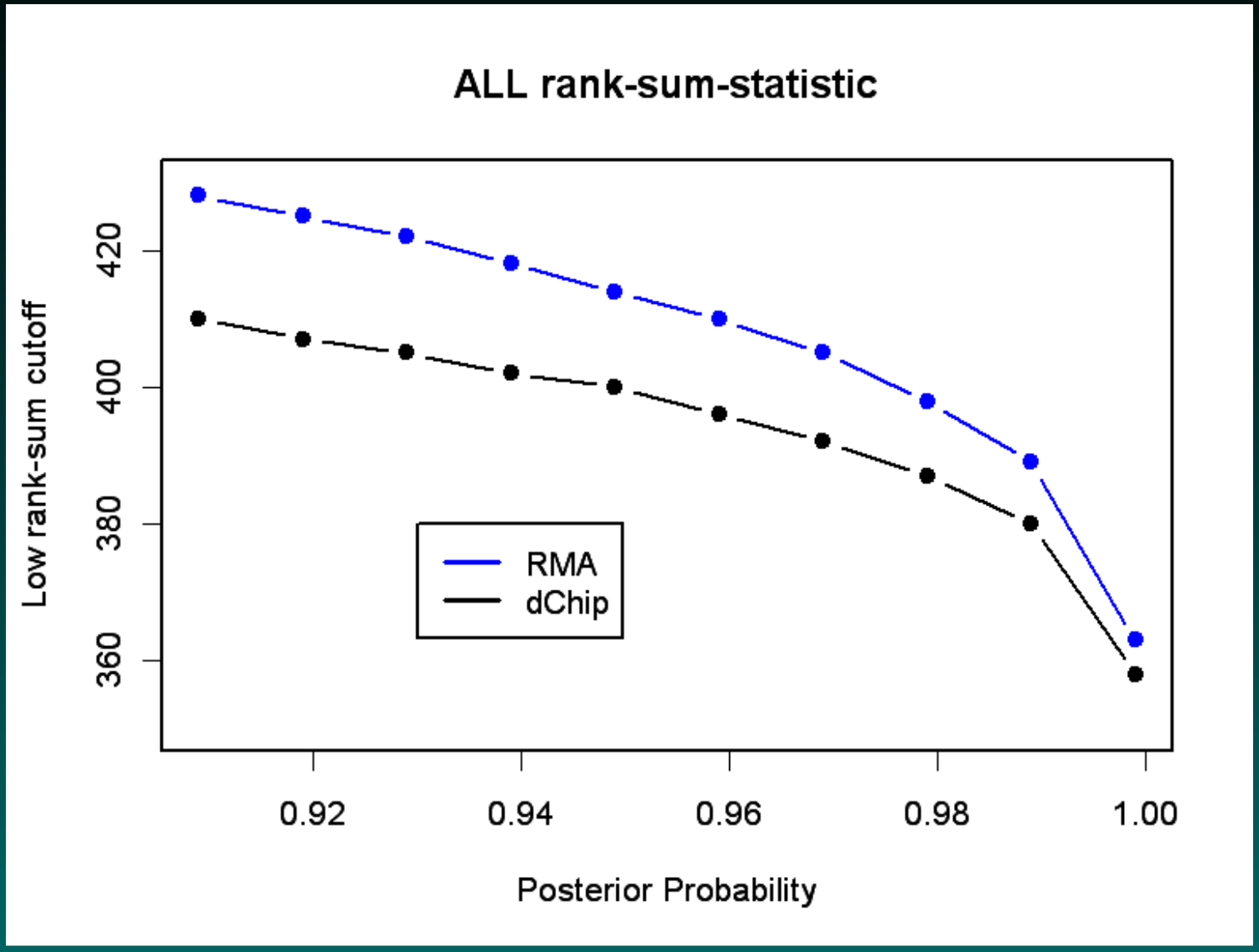
Figure 3
Calibration results for ovary and colon data. The threshold test statistic required to obtain a given FDR_0 for each method is plotted against the FDR_0 value. Results are shown for the colon data (left column), the ovary data (right column), and for the t-test statistic (top row), and the rank-sum statistic (bottom row).

```
> alpha <- seq(0.001, 0.1, by=0.01)
> g <- function(a, data) {
+   pval <- cutoffSignificant(data, alpha=a,
+             by='FDR')
+   qt(1-2*pval, 70)
+ }
> dchip.cut <- sapply(alpha, g, dchip.b)
> rma.cut   <- sapply(alpha, g, rma.b)
> plot(alpha, dchip.cut,
+      xlab='FDR', ylab='t-statistic threshold',
+      main='ALL t-statistic', type='b', pch=16,
+      ylim=c(1,4.15))
> lines(alpha, rma.cut, type='b', pch=16,
+       col='blue')
> legend(0.05, 4, c('RMA', 'dChip'), lwd=3,
+        col=c('blue', 'black'))
```

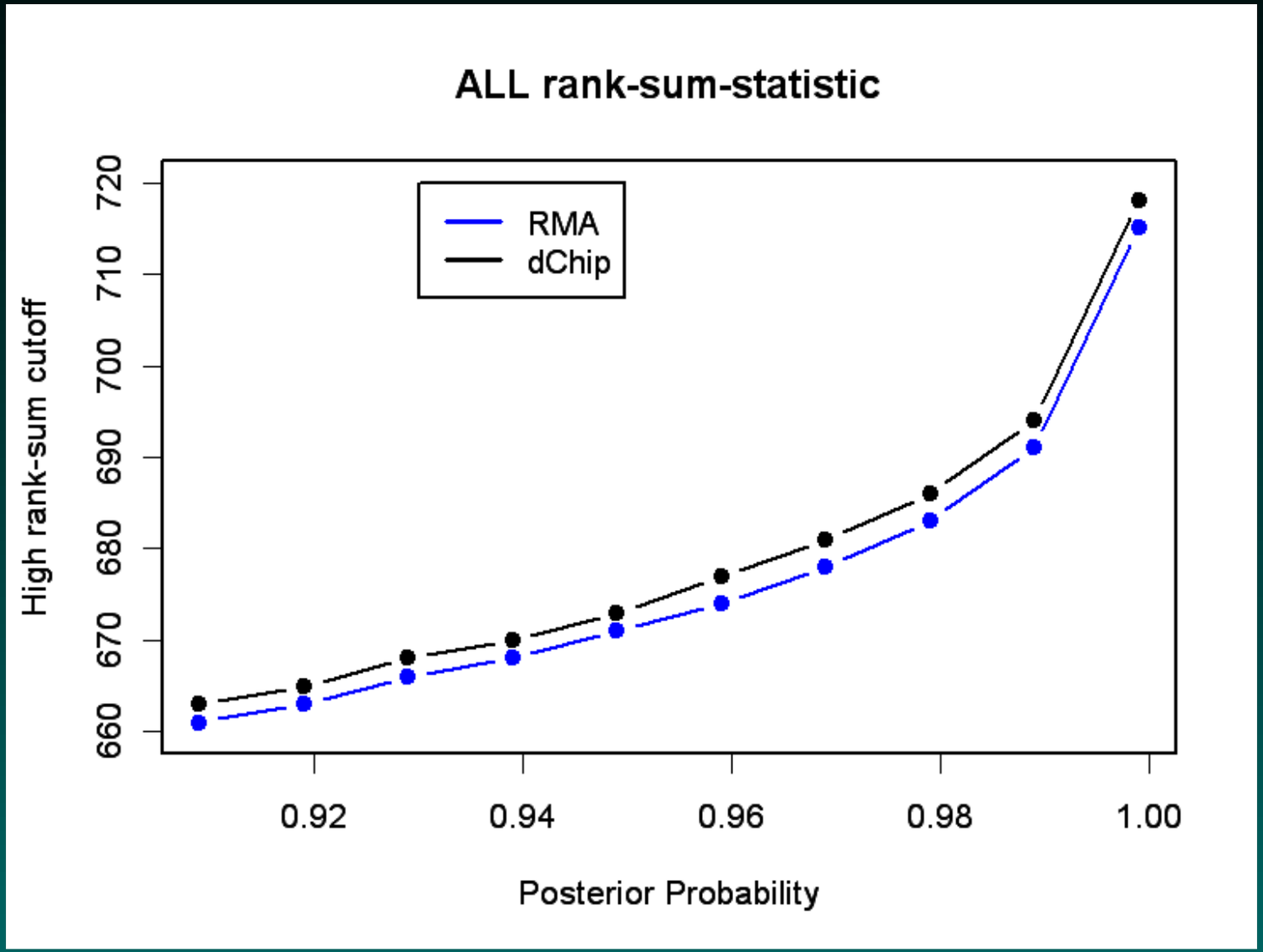
ALL T-Statistic Thresholds



ALL Rank-Sum Thresholds: Low



ALL Rank-Sum Thresholds: High



Calibration determines sensitivity

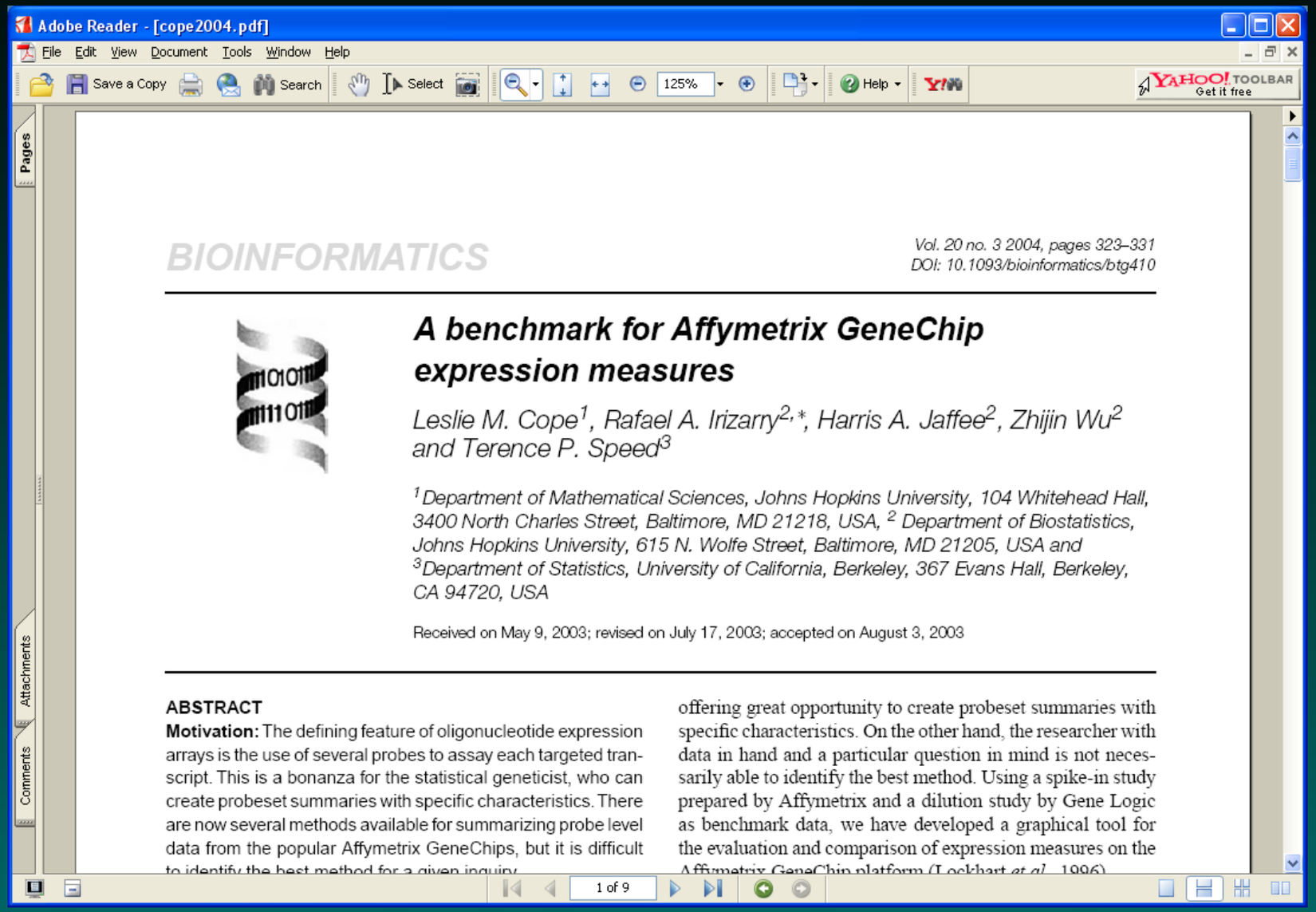
Shedden et al. found the same relation between sensitivity (the number of probe sets called different at a given FDR level) and calibration (the threshold needed to call a probe set different at a given FDR level) in their data sets that we see in our data sets.

Namely, methods that provide greater sensitivity do so by lowering the threshold required to call the statistic significant.

In contrast, they found that dChip and TM consistently performed as well or better than other methods on their two data sets. We, of course, found that RMA is “more sensitive”.

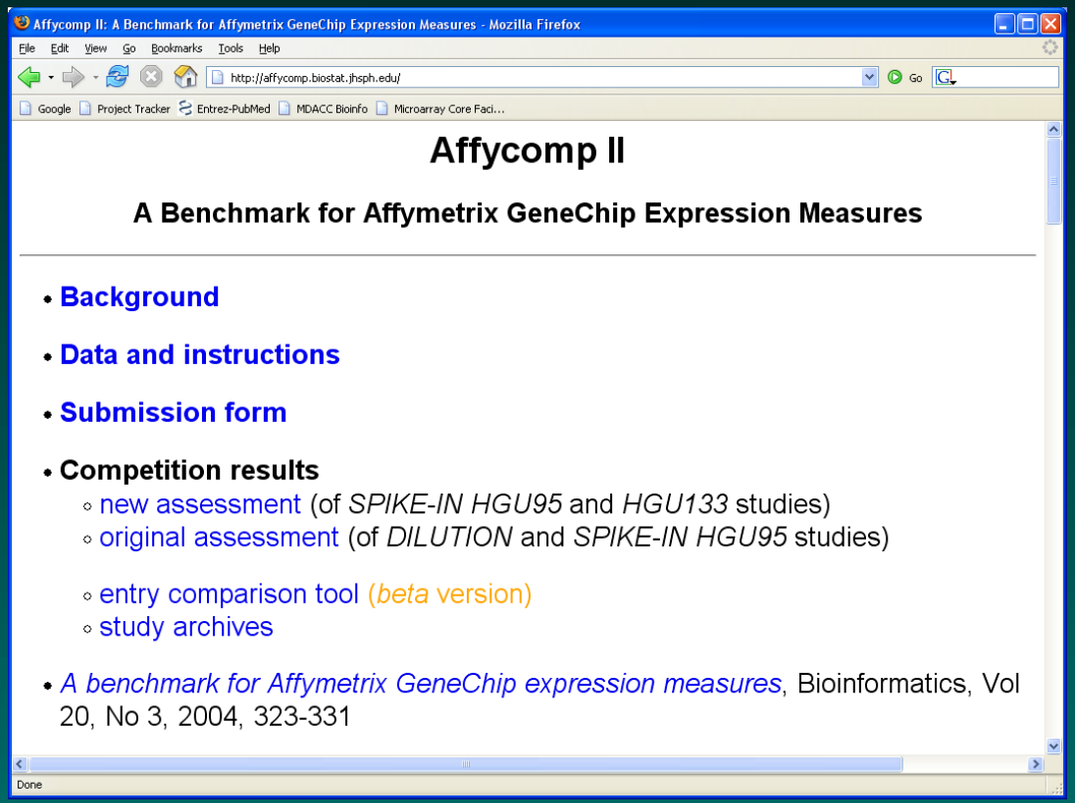
They also found (and we agree) that the choice of processing method has a bigger impact on differntial expression than the choice of using a paramteric t-statistic compared to a non-parametric rank-sum statistic.

Cope et al., Bioinformatics, 2004; 20:323-331



Benchmarking methods using "calibration" data

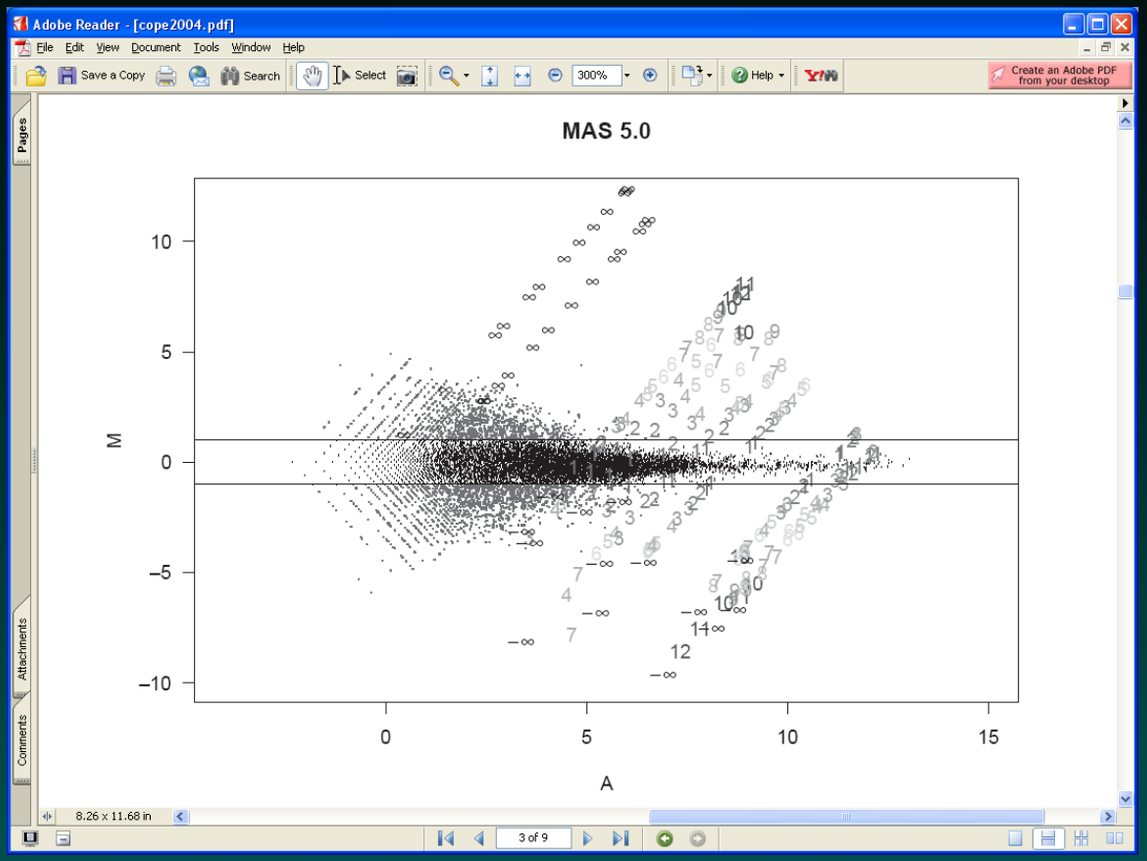
An alternative approach to comparing the results of different processing methods relies on standard sets of spike-in experiments. The performance measures described by Cope et al, are available on a web site:



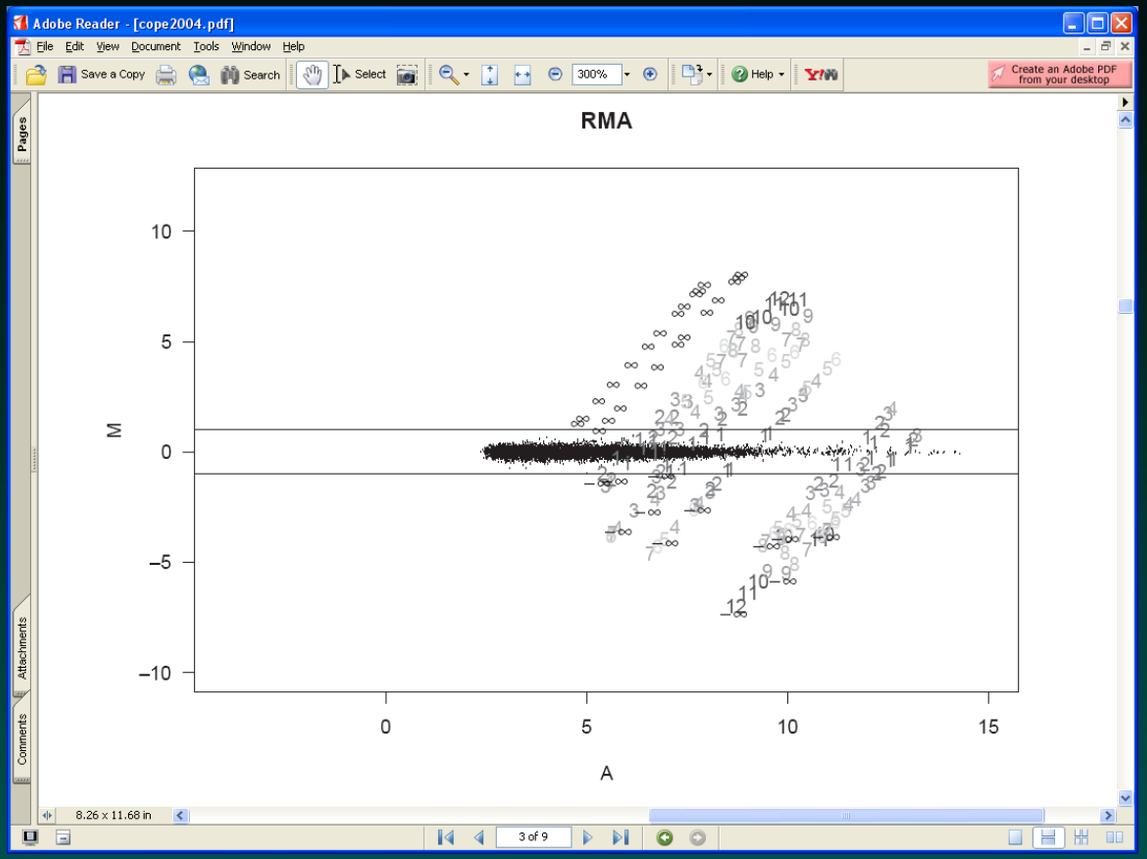
- They use
  - the GeneLogic dilution study that mixed RNA from liver and CNS tissue in different dilutions and proportions
  - the Affymetrix latin-square spike-in study on U95A arrays
  - the Affymetrix latin-square spike-in study on U133A arrays

One should also note that these data sets can be used to evaluate any Affymetrix processing method, by loading the `affycomp` package.

MAplot of Latin-square data: MAS5

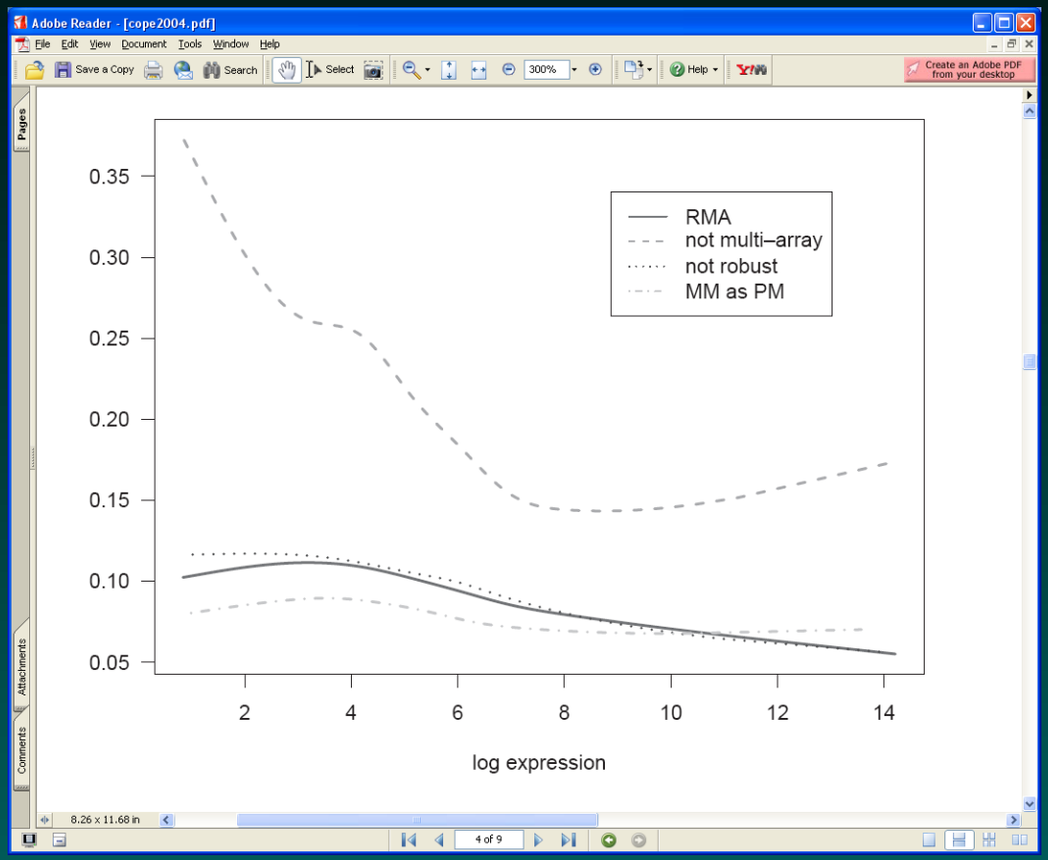


MAplot of Latin-square data: RMA



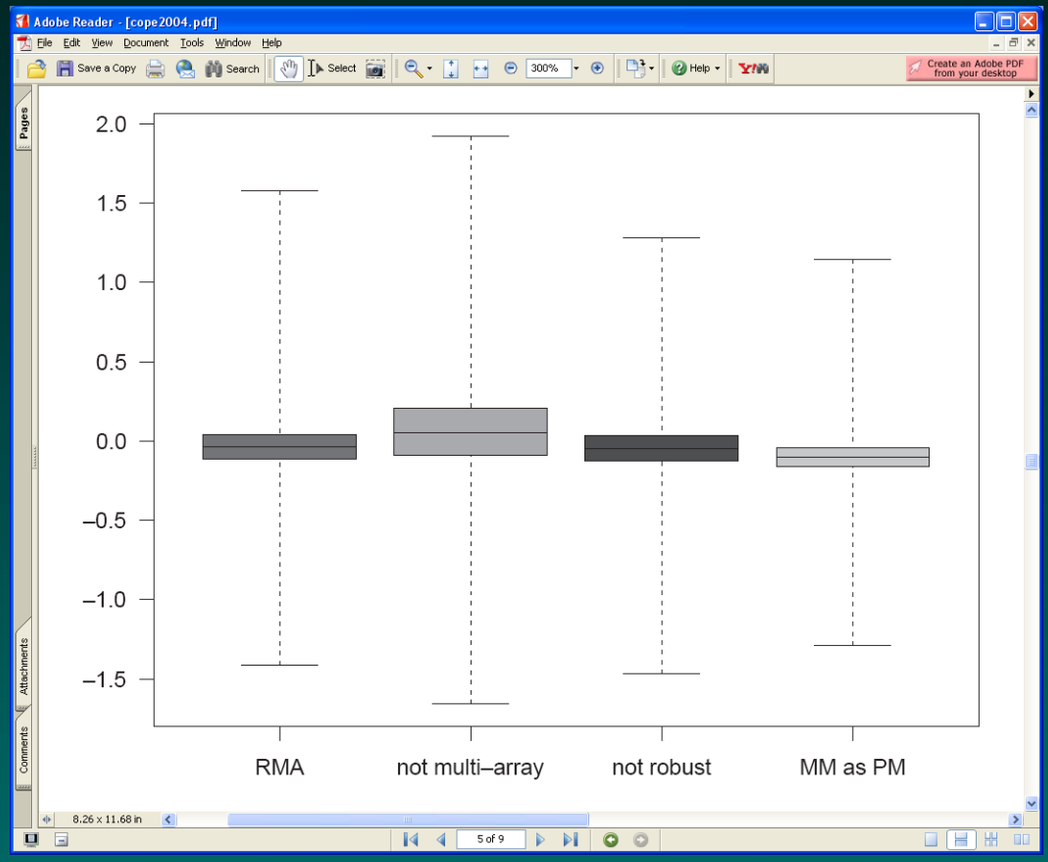
Standard deviation across replicate dilution arrays

This figure shows the results for four different methods.



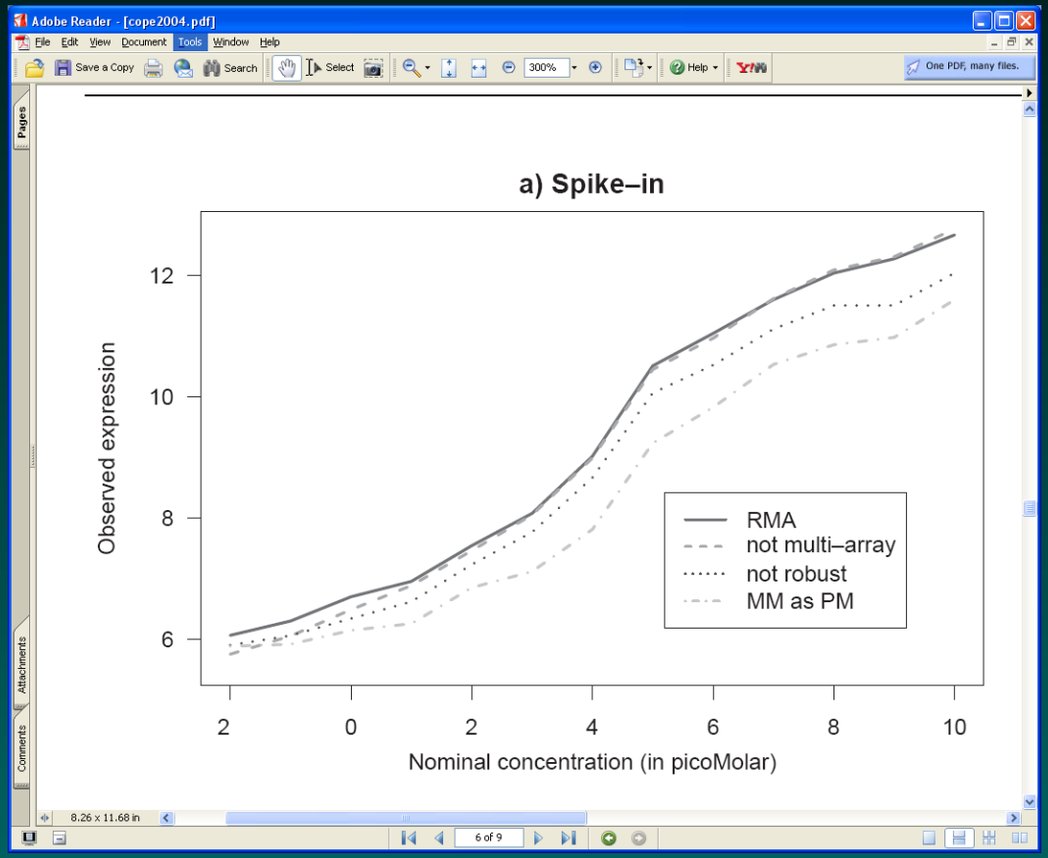
Sensitivity to total amount of RNA

For each method, compute the log ratios (fold changes) between lowest (1.25 μg) and highest (20 μg) concentrations in the dilution experiment.



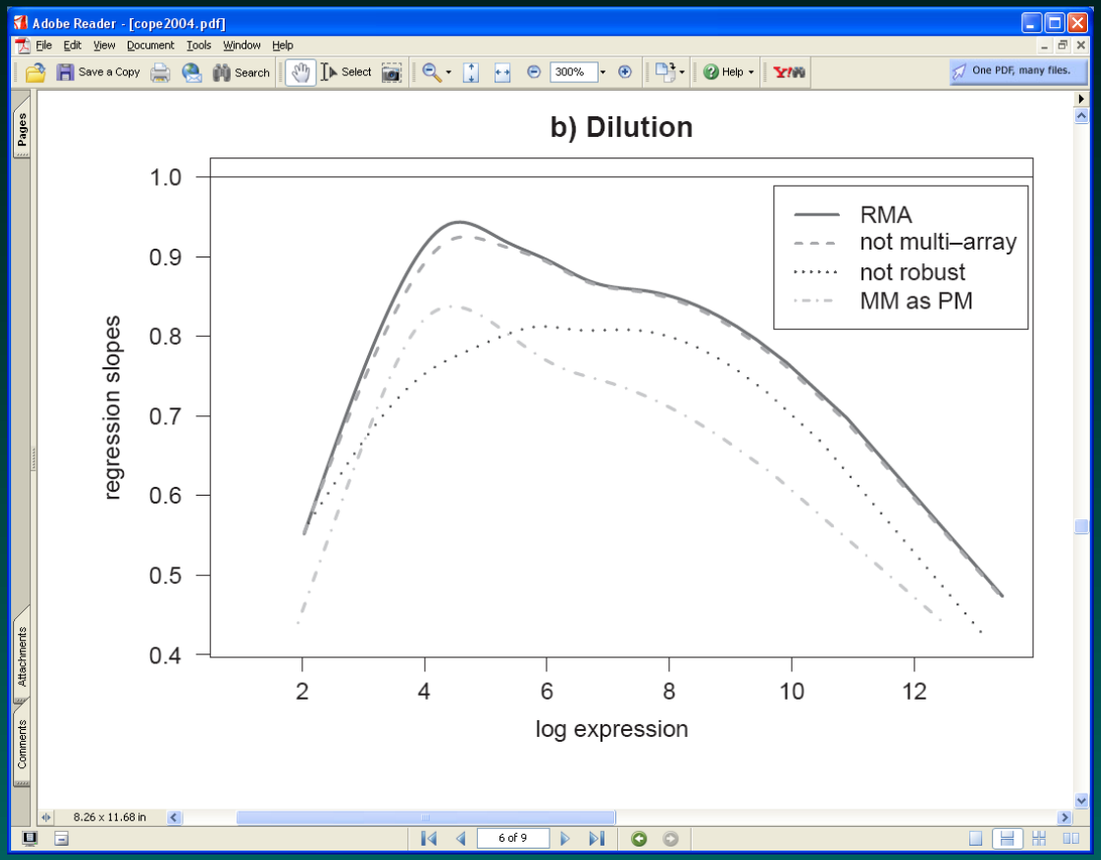
Observed expression vs. nominal concentration in Latin-square

This figure shows the results for four different methods.



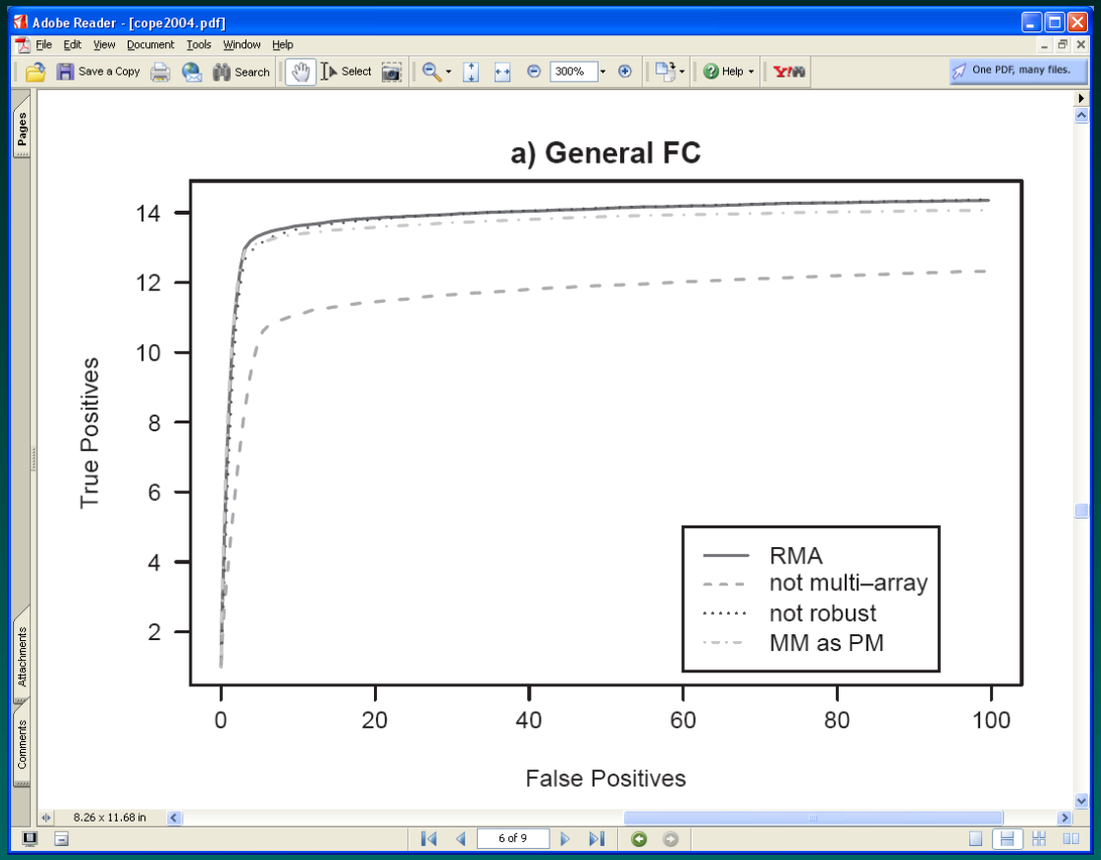
Observed vs. Nominal in Dilution

This figure shows the results for four different methods; they fit regressions to intensity as a function of dilution.



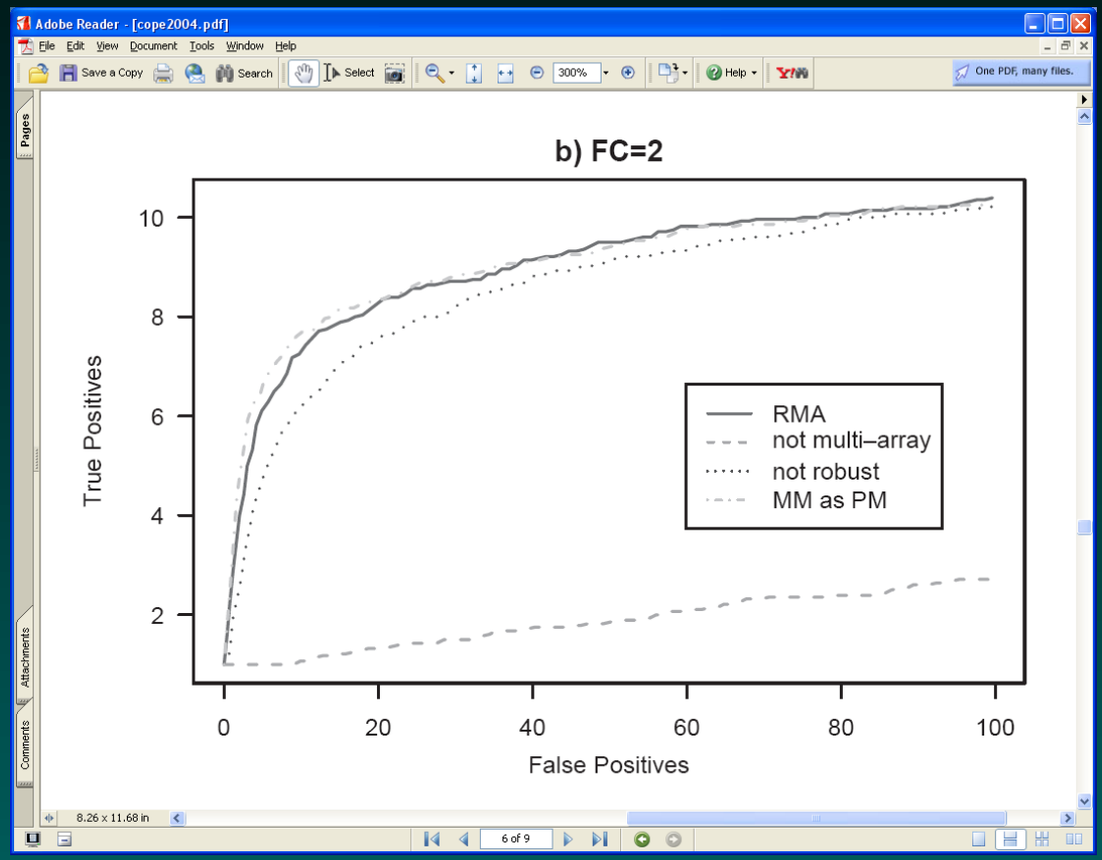
ROC curves: general FC

This figure shows the results for four different methods. In each case, they average the ROC curves for different pairwise comparisons in the Latin-square data.



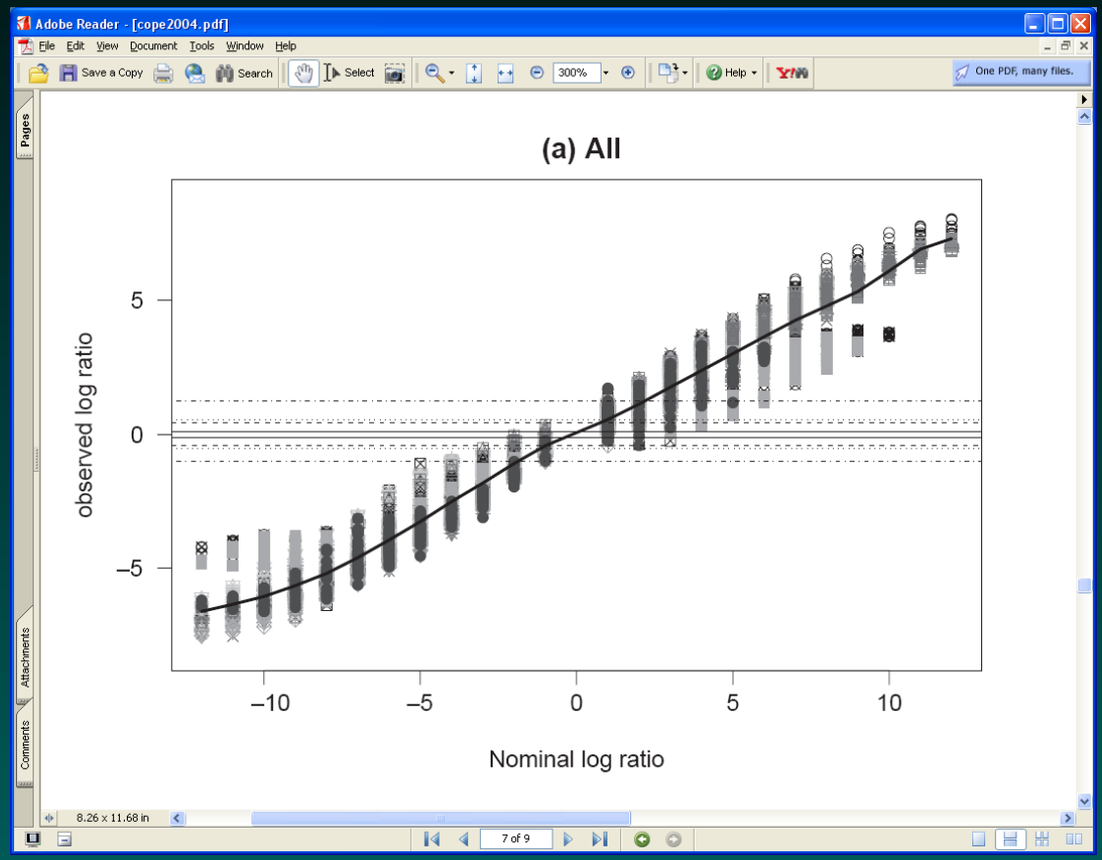
ROC curves: FC=2

This figure shows the results for four different methods



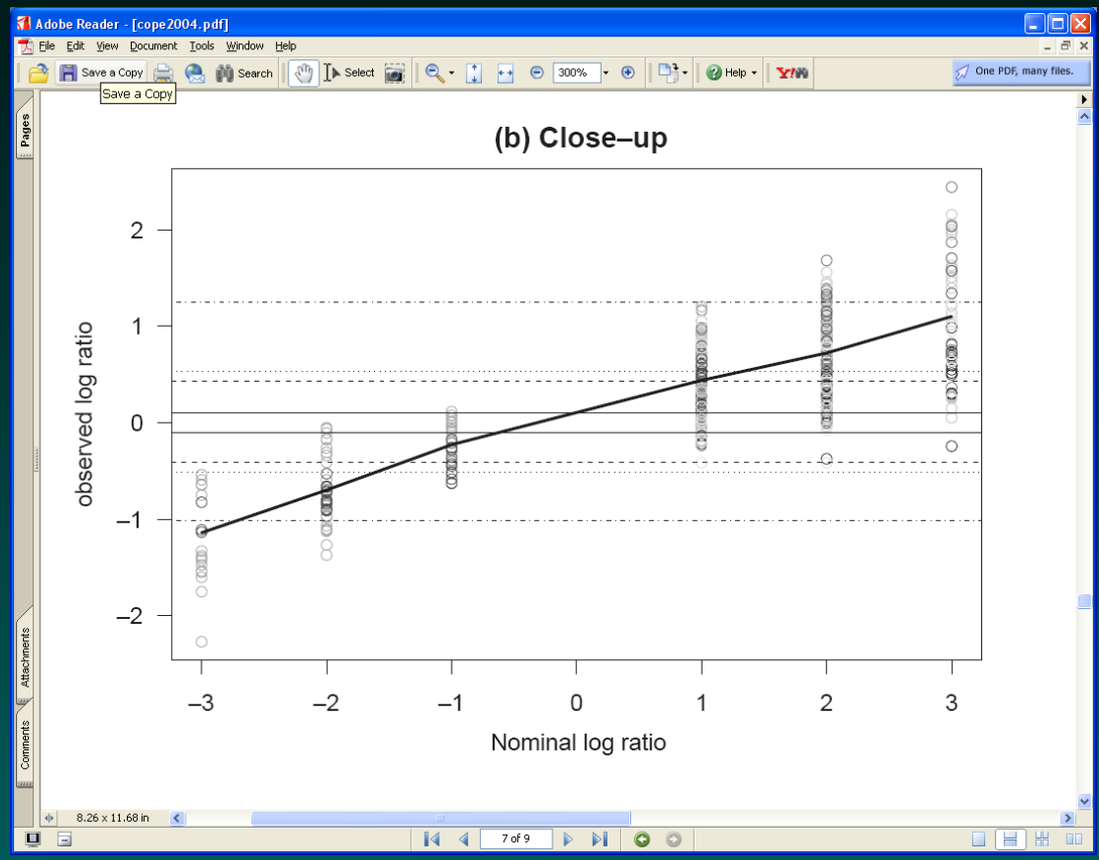
Observed vs. nominal fold change: RMA

This figure shows the results for four different methods



Observed vs. nominal fold change: RMA, close-up

This figure shows the results for four different methods



Measures of performance

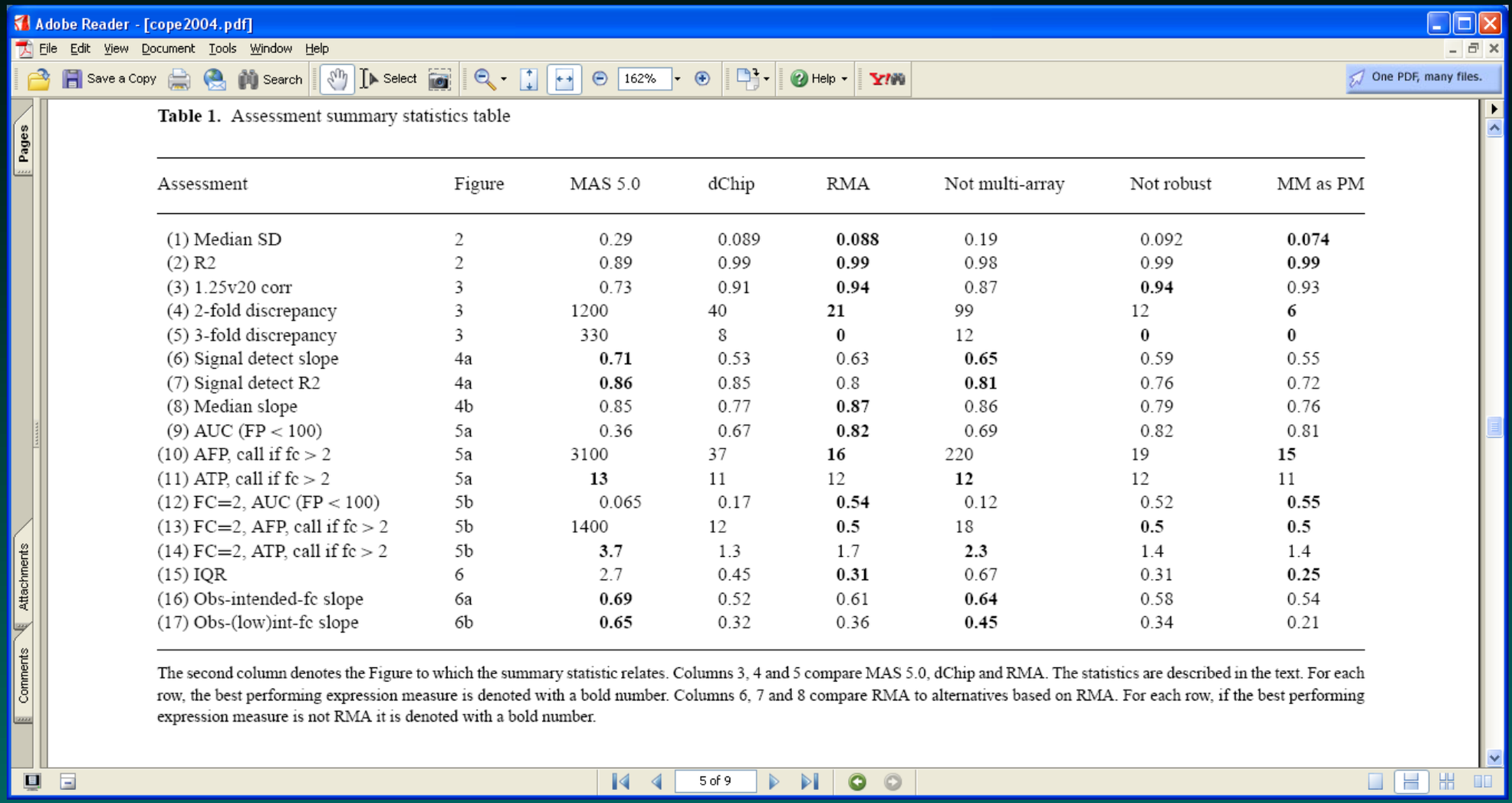


Table 1. Assessment summary statistics table

Assessment	Figure	MAS 5.0	dChip	RMA	Not multi-array	Not robust	MM as PM
(1) Median SD	2	0.29	0.089	**0.088**	0.19	0.092	**0.074**
(2) R2	2	0.89	0.99	**0.99**	0.98	0.99	**0.99**
(3) 1.25v20 corr	3	0.73	0.91	**0.94**	0.87	**0.94**	0.93
(4) 2-fold discrepancy	3	1200	40	**21**	99	12	**6**
(5) 3-fold discrepancy	3	330	8	**0**	12	**0**	**0**
(6) Signal detect slope	4a	**0.71**	0.53	0.63	**0.65**	0.59	0.55
(7) Signal detect R2	4a	**0.86**	0.85	0.8	**0.81**	0.76	0.72
(8) Median slope	4b	0.85	0.77	**0.87**	0.86	0.79	0.76
(9) AUC (FP < 100)	5a	0.36	0.67	**0.82**	0.69	0.82	0.81
(10) AFP, call if fc > 2	5a	3100	37	**16**	220	19	**15**
(11) ATP, call if fc > 2	5a	**13**	11	12	**12**	12	11
(12) FC=2, AUC (FP < 100)	5b	0.065	0.17	**0.54**	0.12	0.52	**0.55**
(13) FC=2, AFP, call if fc > 2	5b	1400	12	**0.5**	18	**0.5**	**0.5**
(14) FC=2, ATP, call if fc > 2	5b	**3.7**	1.3	1.7	**2.3**	1.4	1.4
(15) IQR	6	2.7	0.45	**0.31**	0.67	0.31	**0.25**
(16) Obs-intended-fc slope	6a	**0.69**	0.52	0.61	**0.64**	0.58	0.54
(17) Obs-(low)int-fc slope	6b	**0.65**	0.32	0.36	**0.45**	0.34	0.21

The second column denotes the Figure to which the summary statistic relates. Columns 3, 4 and 5 compare MAS 5.0, dChip and RMA. The statistics are described in the text. For each row, the best performing expression measure is denoted with a bold number. Columns 6, 7 and 8 compare RMA to alternatives based on RMA. For each row, if the best performing expression measure is not RMA it is denoted with a bold number.