A Sample Affymetrix Analysis

Kevin R. Coombes

1 February 2006

1 Loading the Data

1.1 R Libraries

We begin by loading **all** the libraries we will need for this analysis. The `sessionInfo` command serves as critical documentation, since it puts the current versions of the libraries used for the analysis into the final report.

```
> library(affy)
> library(simpleaffy)
> library(geneplotter)
> library(colorspace)
> library(xtable)
> library(ClassComparison)
> library(ClassDiscovery)
> sessionInfo()
```

```
R version 2.4.0 (2006-10-03)
i386-pc-mingw32

locale:
LC_COLLATE=English_United States.1252;LC_CTYPE=English_United States.1252;LC_MONETARY=English_United Sta

attached base packages:
[1] "splines"   "tools"     "methods"   "stats"     "graphics"  "grDevices"
[7] "utils"     "datasets"  "base"

other attached packages:
 hgu133plus2cdf  ClassDiscovery         cluster ClassComparison      PreProcess
       "1.14.0"           "1.3"        "1.11.2"           "1.3"           "1.3"
      oompaBase          xtable      colorspace     geneplotter        annotate
          "1.3"         "1.4-2"           "0.9"        "1.12.0"        "1.12.0"
     simpleaffy      genefilter        survival            affy          affyio
        "2.8.0"        "1.12.0"          "2.29"        "1.12.0"         "1.2.0"
        Biobase
       "1.12.2"
```

1.2 Experiment Description: The MIAME object

A MIAME object records general information about the experiment. The MIAME object should be constructed at the beginning of the analysis. In an ideal world, this information would come from a database or an XML file. Here, we type it in by hand; since this contains information specific to an individual project, we cannot put it into a separate reusable module.

```
> miame <- new("MIAME", name = "Toru Nakamura", lab = "I.J. Fidler",
+     contact = "Toru Nakamura <TNakamur@mdanderson.org>", title = "Peripheral and central zones of panc
+     abstract = "Differential expression of genes in tumor cells growing in the peripheral and central
+     other = list(Nothing = "No other information is available"))
```

1.3 Using MIAME objects

The next step is not necessary in most reports; it is here to demonstrate how to use MIAME objects. Accessing the information in the MIAME object is straightforward.

```
> miame

Experiment data
  Experimenter name: Toru Nakamura
  Laboratory: I.J. Fidler
  Contact information: Toru Nakamura <TNakamur@mdanderson.org>
  Title: Peripheral and central zones of pancreatic cancer
  URL:
  PMIDs:

  Abstract: A 24 word abstract is available. Use 'abstract' method.
```

Note that the abstract is not displayed by default. You use the abstract command to extract it. Since most abstracts are long, it is useful to present it in a way that will look better in the final report.

```
> writeLines(strwrap(abstract(miame)))

Differential expression of genes in tumor cells growing in the
peripheral and central zones of pancreatic cancer was measured using
Affymetrix U133 Plus2.0 arrays.
```

All MIAME objects should include a list of “other” items, which are accessed using the notes function. Some of the auxiliary packages in BioConductor that display results will not work properly if this item is omitted.

```
> notes(miame)

$Nothing
[1] "No other information is available"
```

1.4 Clinical Information: the phenoData object

In order to perform an analysis of microarray data, we need to know something about the samples that were hybridized to the microarrays. In standard R usage, this sort of "clinical" information is typically stored in a data frame. In the BioConductor approach, it is stored in a `phenoData` object or in an `AnnotatedDataFrame`. Both kinds of object represent a data frame that comes packaged with instructions for how to interpret each one of the columns. In current (and older) versions of BioConductor, we need to use `phenoData`; the plan for future versions is to convert to using an `AnnotatedDataFrame`. We illustrate how to produce both kinds.

In the example used here, the only thing that matters about each sample is whether it came from the periphery or the center of a tumor. Note that this is actually matched data, so the pairing matters. This information was stored in an external file in tab-separated-values format, which we read in using the usual `read.table` contortions. Note that this sample information file follows the conventions required by dChip. In particular, the first column is called "ArrayName" and contains the CEL file name (without the extension) and the second column is called "SampleName" and contains a short name that should be used when displaying information about the sample.

```
> si <- read.table("sampleInfo.txt", sep = "\t", header = TRUE)
> rownames(si) <- si[, "SampleName"]
```

It is useful to display this information in a table inside a report; we use the `xtable` command from the `xtable` package for this purpose. You can use `xtable` to produce tables in LaTeX format for direct inclusion in Sweave reports, or you can use it to produce tables in HTML format that can be made available separately for investigators to import into Word documents. In our case, the CEL file names generated by the Affymetrix Core Facility include underscore characters, which are not processed correctly by LaTeX, so we have to go through a few contortions to manipulate them and get a usable result.

```
> xtab <- xtable(si, label = "tbl:sample", caption = "Properties of the samples hybridized to each cel f
> print(xtab, type = "html", file = "table1.html")
> temp <- print(xtab)
> temp <- gsub("_", "$\\\\_$", temp)

> writeLines(temp)
```

	ArrayName	SampleName	Location	TumorID
Cen1	U133_Plus_2_171-IF-007_08_21_06	Cen1	Center	1.00
Cen2	U133_Plus_2_171-IF-008_08_21_06	Cen2	Center	2.00
Cen3	U133_Plus_2_171-IF-009_08_21_06	Cen3	Center	3.00
Per1	U133_Plus_2_171-IF-010_08_21_06	Per1	Periphery	1.00
Per2	U133_Plus_2_171-IF-011_08_21_06	Per2	Periphery	2.00
Per3	U133_Plus_2_171-IF-012_08_21_06	Per3	Periphery	3.00

Table 1: Properties of the samples hybridized to each cel file.

Because the interpretations of the columns are not at present stored in some easy-to-read file, we also have to construct them. The `AnnotatedDataFrame` class wants them in a data frame that at least includes a column called `labelDescription`; the `phenoData` objects wants them in a list.

```
> vl <- list(ArrayName = "Name of the CEL file, without extension",
+     SampleName = "Short sample name", Location = "Location within the pancreatic tumor",
+     TumorID = "Unique identifier for each tumor")
> vmd <- data.frame(labelDescription = unlist(vl))
```

Now we can put the explanations together with the clinical data:

```
> adf <- new("AnnotatedDataFrame", data = si, varMetadata = vmd)
> pd <- new("phenoData", pData = si, varLabels = vl, varMetadata = vmd)
> rm(si, vmd, vl)
```

Here is a check that this makes sense:

```
> pd

        phenoData object with 4 variables and 6 cases
        varLabels
                ArrayName: Name of the CEL file, without extension
                SampleName: Short sample name
                Location: Location within the pancreatic tumor
                TumorID: Unique identifier for each tumor

> pData(pd)

                           ArrayName SampleName  Location TumorID
Cen1 U133_Plus_2_171-IF-007_08_21_06       Cen1    Center       1
Cen2 U133_Plus_2_171-IF-008_08_21_06       Cen2    Center       2
Cen3 U133_Plus_2_171-IF-009_08_21_06       Cen3    Center       3
Per1 U133_Plus_2_171-IF-010_08_21_06       Per1 Periphery       1
Per2 U133_Plus_2_171-IF-011_08_21_06       Per2 Periphery       2
Per3 U133_Plus_2_171-IF-012_08_21_06       Per3 Periphery       3
```

1.5 Reading the CEL files: an AffyBatch

Now we can actually read the cel files, using the `read.affybatch` command. For small projects, this will be the prefered method. However, there are serious limits to the number of cel files that can be used in their raw form on a 32-bit machine. You are limited to fewer than 30 of the newest arrays (such as the U133A Plus 2.0 human arrays) or to fewer that about 90 of the older U95A arrays.

```
> fnames <- paste(as.character(pData(pd)$ArrayName), "cel", sep = ".")
> ab <- read.affybatch(filenames = fnames, phenoData = pd, description = miame)
> rm(fnames)
> ab

AffyBatch object
size of arrays=1164x1164 features (63520 kb)
cdf=HG-U133_Plus_2 (54675 affyids)
number of samples=6
number of genes=54675
annotation=hgu133plus2
```

Note that the `phenoData` object, `pd`, and the `MIAME` object, `miame` were both passed into the constructor, so the resulting `AffyBatch` object, `ab`, now includes that information. When you first display the object, the system automatically checks that you have an up-to-date annotation library for the kinds of arrays that were used. In this case, the annotations are contained in the `hgu133plus2` library.

2 Basic Quality Control

We use the `simpleaffy` package for initial QC. In order to gather the statistics, we have to process the CEL files using the MAS5 algorithm, and then use the `qc` function. (If you do not call the MAS5 algorithm first, then it will be invoked automatically as part of the `qc` call.)

```
> x.mas <- call.exprs(ab, algorithm = "mas5")
```

```
Background correcting
Retrieving data from AffyBatch...done.
Computing expression calls...
......done.
scaling to a TGT of 100 ...
Scale factor for: Cen1 1.08417090632798
Scale factor for: Cen2 1.13659569486376
Scale factor for: Cen3 2.27283458528825
Scale factor for: Per1 0.539723974222116
Scale factor for: Per2 5.017651182079
Scale factor for: Per3 0.848903847544313
```

```
> x.qc <- qc(ab, x.mas)
```

```
Getting probe level data...
Computing p-values
Doing PMA Calls
```

2.1 Summary QC Statistics

In our experience, the best relative measure of overall quality is provided by the "percent present" calls. We typically expect to see the percentage of genes called "present" roughly in the range of 30% to 60%; the exact values differ base one type of array and the type of tissue. Wherever they lie in the range, they should be consistent (ideally, with a spread of less than ten percentage points) across all the arrays in the experiment.

```
> percent.present(x.qc)
```

```
Cen1.present Cen2.present Cen3.present Per1.present Per2.present Per3.present
    40.60357     38.79104     34.24783     39.67627     27.05075     38.33562
```

The average, minimum, and maximum backgrounds should be similar across the arrays, varying less than twofold.

```
> avbg(x.qc)
```

```
    Cen1     Cen2     Cen3     Per1     Per2     Per3
39.28419 42.78025 38.60290 63.86891 32.14879 48.93674
```

```
> minbg(x.qc)
```

```
    Cen1     Cen2     Cen3     Per1     Per2     Per3
38.08860 41.15040 36.25201 60.35298 31.29746 45.97613
```

```
> maxbg(x.qc)

    Cen1     Cen2     Cen3     Per1     Per2     Per3 
41.88270 44.91899 41.37978 69.10045 33.88205 53.13545 
```

Affymetrix recommends that the scaling factors used to bring the arrays onto a similar scale should not differ by more than about threefold. The scaling factors are based on a trimmed mean intensity; large differences in scaling factors may indicate either low quality RNA or situations where the normalization assumptions may not hold.

```
> sfs(x.qc)

[1] 1.0841709 1.1365957 2.2728346 0.5397240 5.0176512 0.8489038
```

Affymetrix includes QC probes targeted to both ends and to the middle of the beta actin and GAPDH genes. The intensity ratios are a measurement of RNA quality, since ratios that are far from one indicate the presence of truncated transcripts.

```
> ratios(x.qc)

     AFFX-HSAC07/X00351.3'/5' AFFX-HUMGAPDH/M33197.3'/5'
Cen1                 5.436416                   2.328079
Cen2                 6.095361                   3.244753
Cen3                 5.858997                   3.357722
Per1                 4.660369                   2.466161
Per2                 5.647859                   3.568095
Per3                 4.554761                   2.539702
     AFFX-HSAC07/X00351.3'/M AFFX-HUMGAPDH/M33197.3'/M
Cen1                2.543474                 0.9920248
Cen2                3.517353                 1.4939736
Cen3                3.308929                 1.2866041
Per1                1.906734                 0.9893949
Per2                3.397787                 1.7794990
Per3                1.646996                 1.0612092
```

Many of the important QC statistics are summarized in a single plot (Figure 1). The scaling factors are displayed extending from a central axis on a blue background; the percent present and average background are listed along the left side; and the 3'/5' ratios for beta actin (triangles) and GAPDH (circles) aer plotted to the right.

2.2 QC Interpretation

In this example, the fewest present calls (27%) are made on the fifth array (Per2). The overall range of calls is fairly consistent, but does exceed ten percent. In addition, the fifth array (Per2) is the dimmest overall, with the fourth (Per1) being about two-fold brighter. There is also a tenfold difference in the scaling factors between the dimmest (Per2) and brightest (Per1) arrays, and a fourfold difference between arrays Cen3 and Per1. Finally, most of the ratios appear to be on the high side.

Additional assessments of the data will use colors to distinguish the different arrays. We prepare a standard vector of color assignments here, stored in an object that will be used by those plotting routines.

```
> colorSet <- hcl(seq(0, 360, length = 7)[1:6], c = 65, l = 65)
```

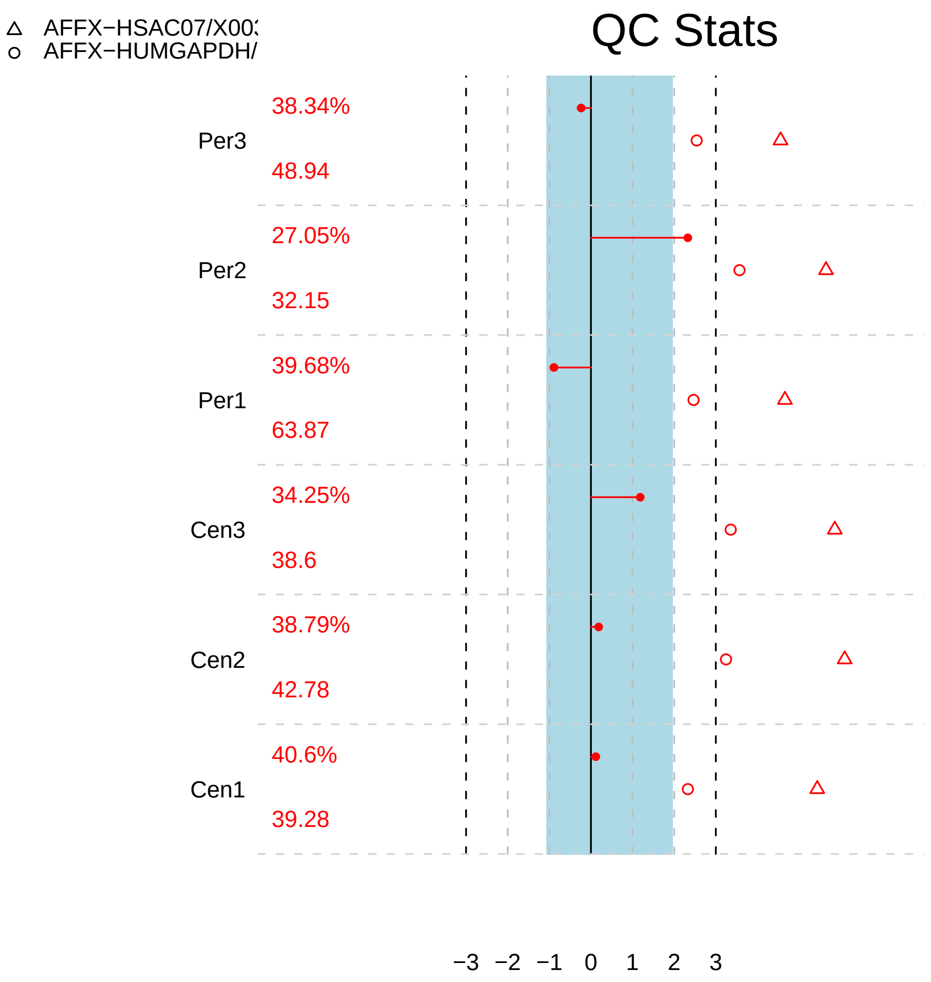


Figure 1: Summary plot of Affymetrix QC statistics.

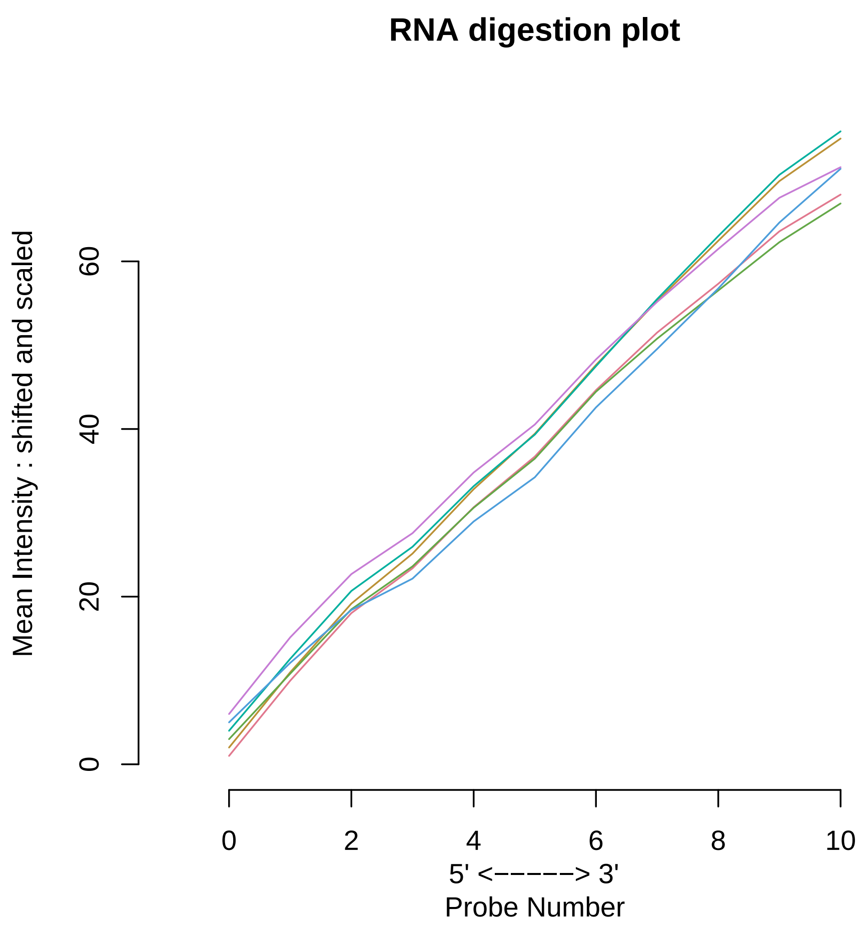


Figure 2: Plot of the mean log intensity as a function of probe position from 3' to 5' end of the target mRNA.

2.3 Checking for RNA Degradation

We can assess the potential for RNA degradation possibility using part of the `affy` package. The summary reports whether the slopes are nonzero, which would indicate some level of degradation. At the very least, one would like all the slopes to be similar, which would indicate that all samples were handled in the same manner. Similar behavior can also be checked in a plot of the average intensity against position (Figure 2).

```
> ard <- AffyRNAdeg(ab)
> summaryAffyRNAdeg(ard)
```

```
            Cen1      Cen2      Cen3      Per1      Per2      Per3
slope  6.70e+00 7.30e+00 6.43e+00 7.17e+00 6.58e+00 6.56e+00
pvalue 1.59e-12 3.68e-14 4.50e-13 4.83e-14 6.30e-12 2.10e-12
```

```
> MAplot(ab[, sample(ncol(exprs(ab)), 3)], pairs = TRUE, plot.method = "smoothScatter")
```

Figure 3: Pairwise Bland-Altman (M-vs-A) plots of the probe-level intensity data for three randomly selected arrays.

2.4 Distributional plots

The `affy` package also contains some plotting routines that help us decide whether normalization is needed and, if so, whether it is behaving sensibly. We start with Bland-Altman (M-versus-A) pairwise plots of some randomly selected arrays (Figure 3). Using the `smoothScatter` method is highly recommended, since it provides much more efficient plotting routines. The need for normalization can also be assessed using density plots (Figure 4, which are produced with `hist` instead of `density`) and boxplots (Figure 5).

3 RMA Quantifications

As part of the standard QC analysis, the CEL files were quantified using the MAS5 algorithm. This method works one array at a time, and so the results it produces may not be as reliable as those from a method like Robust Multiarray Analysis (RMA) that borrows strength across arrays. The `rma` function applies the RMA algorithm to quantify the Affymetrix data.

Note: in the current implementation of BioConductor, both x.mas and x.rma are returned as objects of class `exprSet`. That class will be deprecated starting with the next release, and will be replaced with the newer class called an rcodeExpressionSet. Our current standard converts `exprSet`s into `ExpressionSet`s so that our downstream analysis can rely on being able to use the newer kind of object.

```
> x.rma <- as(rma(ab), "ExpressionSet")

Background correcting
Normalizing
Calculating Expression

> x.mas <- as(x.mas, "ExpressionSet")
```

In Figure 6, we produce boxplots of the log intensity of probe-set expression computed using the two different quantification algorithms. It is often the case that MAS5 data requires additional normalization across arrays to get the distributions to match.

4 Differential Expression

We are almost reasdy to perform ana analysis to detect differntialy expressed genes. In the first analysis, we are going to ignore the paired nature of the samples, in part to illustrate the difficulties involved in analyzing data sets that only include three samples in each group.

We will analyze three different data sets:

1. The RMA quantifications
2. The MAS5 quantifications
3. Median-normalized MAS5 quantifications

```
> hist(ab, col = colorSet, lty = 1)
```

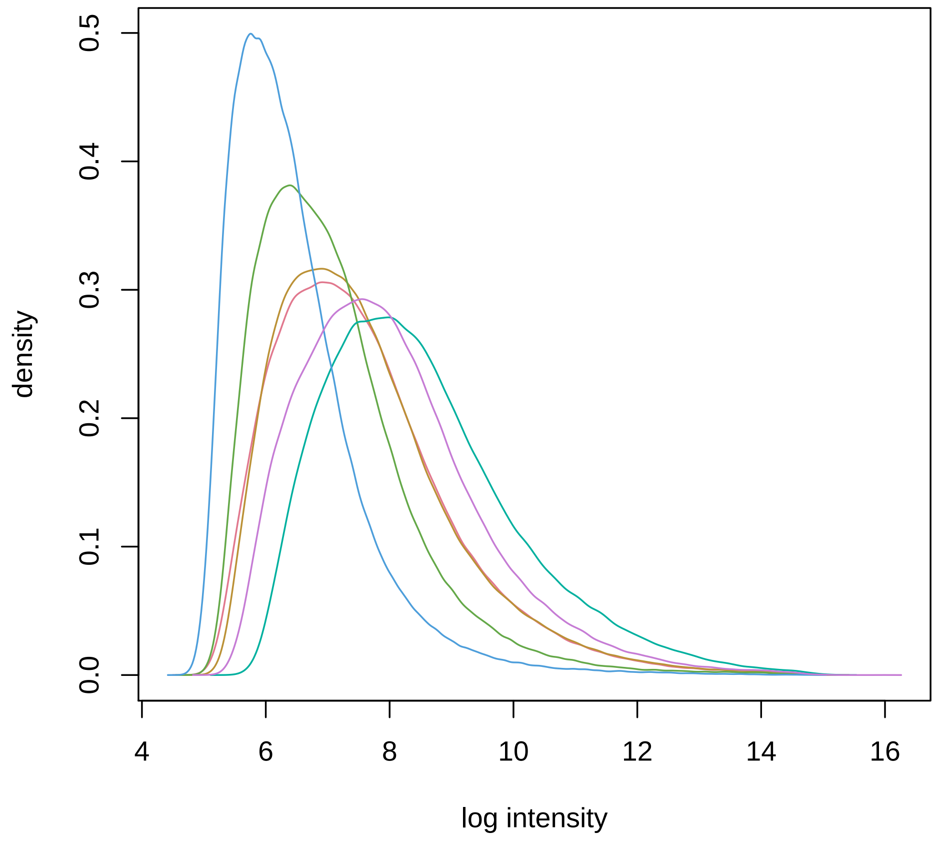


Figure 4: Density plots of the probe-level log intensity data in each array.

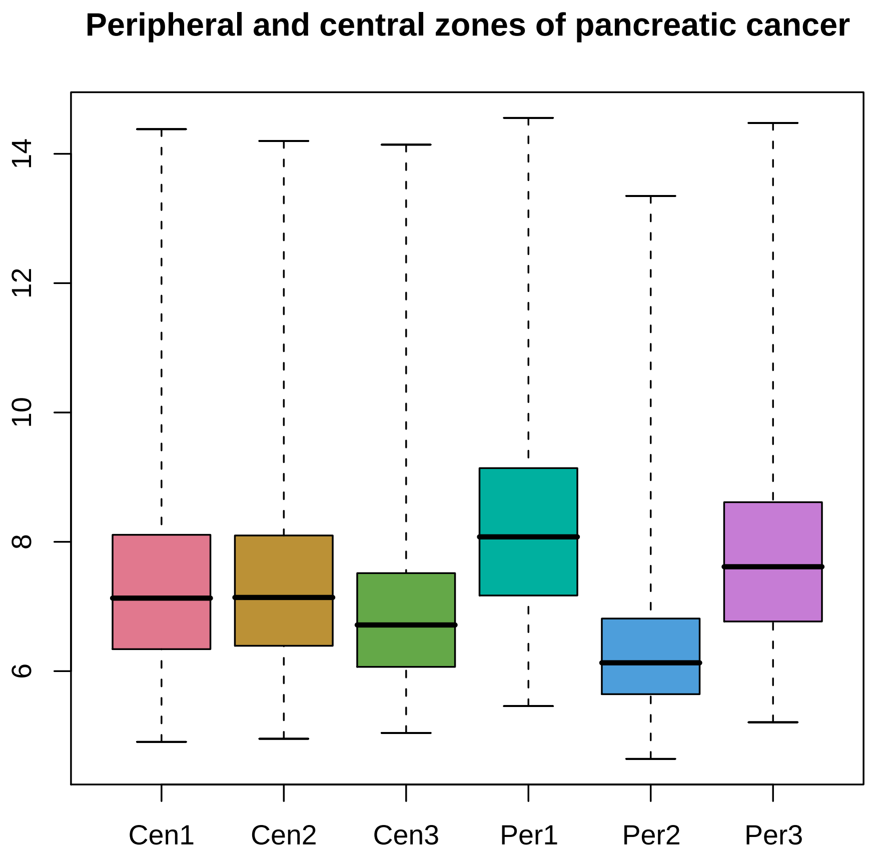
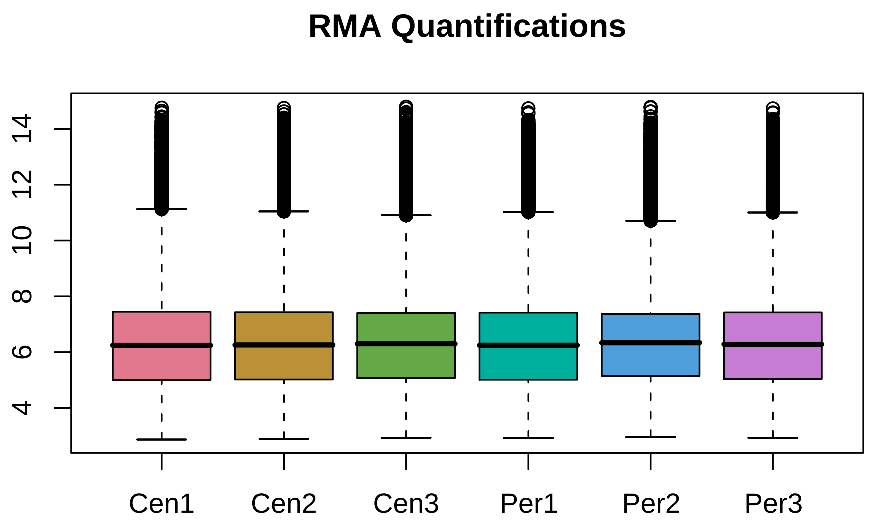


Figure 5: Box plots of the probe-level log intensity data in each array.



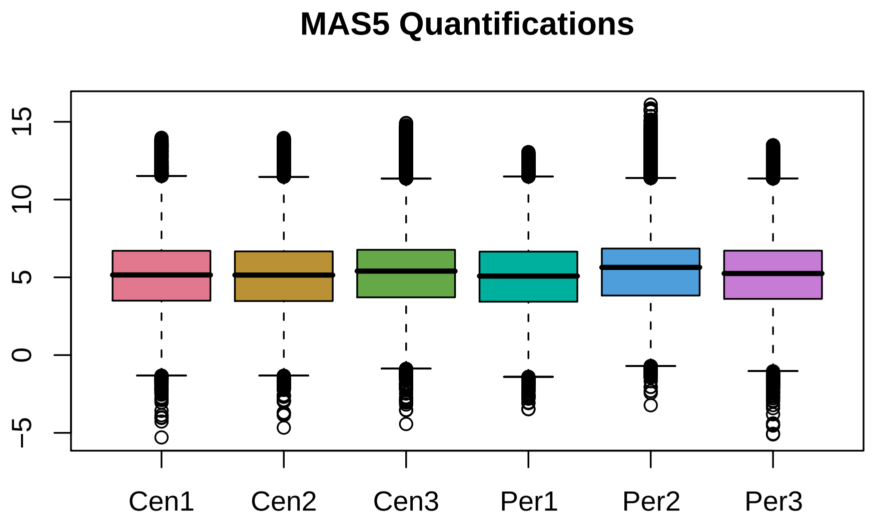


Figure 6: Box plots of the log intensity distributions of quantifications oibtained using the RMA algorithm (top) and the MAS5 algorithm (bottom).

As seen from the boxplots in Figure 6, the medians of the expressed data after MAS5 quantification are not aligned. In order to correct this, we create a new object and manually align the medians.

```
> x.mas5med <- x.mas
> data <- exprs(x.mas)
> exprs(x.mas5med) <- sweep(data, 2, apply(data, 2, median) - 5,
+     "-")
```

4.1 Differential Expression using RMA

The `MultiTtest` function performs gene-by-gene two-sample t-tests. The second argument is a character string indicating which factor in the phenoData object should be used for the comparison.

```
> rma.mtt <- MultiTtest(x.rma, "Location")
> summary(rma.mtt)
```

```
Row-by-row two-sample t-tests with 54675 rows
Positive sign indicates an increase in class: Center

Call: MultiTtest(data = x.rma, classes = "Location")

T-statistics:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-35.0500  -0.9888  -0.2063  -0.1016   0.6813  36.5000

P-values:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
3.364e-06 1.986e-01 4.396e-01 4.577e-01 7.079e-01 1.000e+00
```

The summary shows that there are some extremely large t-statistics. It also points out that positive t-statistics indicate overexpression of a gene in the center of the tumor, so negative t-statistics indicate higher expression in the periphery of the tumor.

TO adjust for multiple testing, we use a beta-uniform mixture (BUM) model to estimate the false discovery rate (FDR). A plot of the fitted model shows that there are definitely some differentially expressed genes, but also suggests that the false discovery rates are likely to be fairly high even at very small p-value cutoffs (Figure 7).

```
> rma.bum <- Bum(rma.mtt@p.values)
> rma.sel <- selectSignificant(rma.bum, alpha = 0.25, by = "FDR")
> countSignificant(rma.bum, alpha = 0.25, by = "FDR")
```

```
[1] 198
```

4.2 Differential Expression using MAS5

The `MultiTtest` function performs gene-by-gene two-sample t-tests. The second argument is a character string indicating which factor in the phenoData object should be used for the comparison.

```
> mas5.mtt <- MultiTtest(x.mas, "Location")
> summary(mas5.mtt)
```

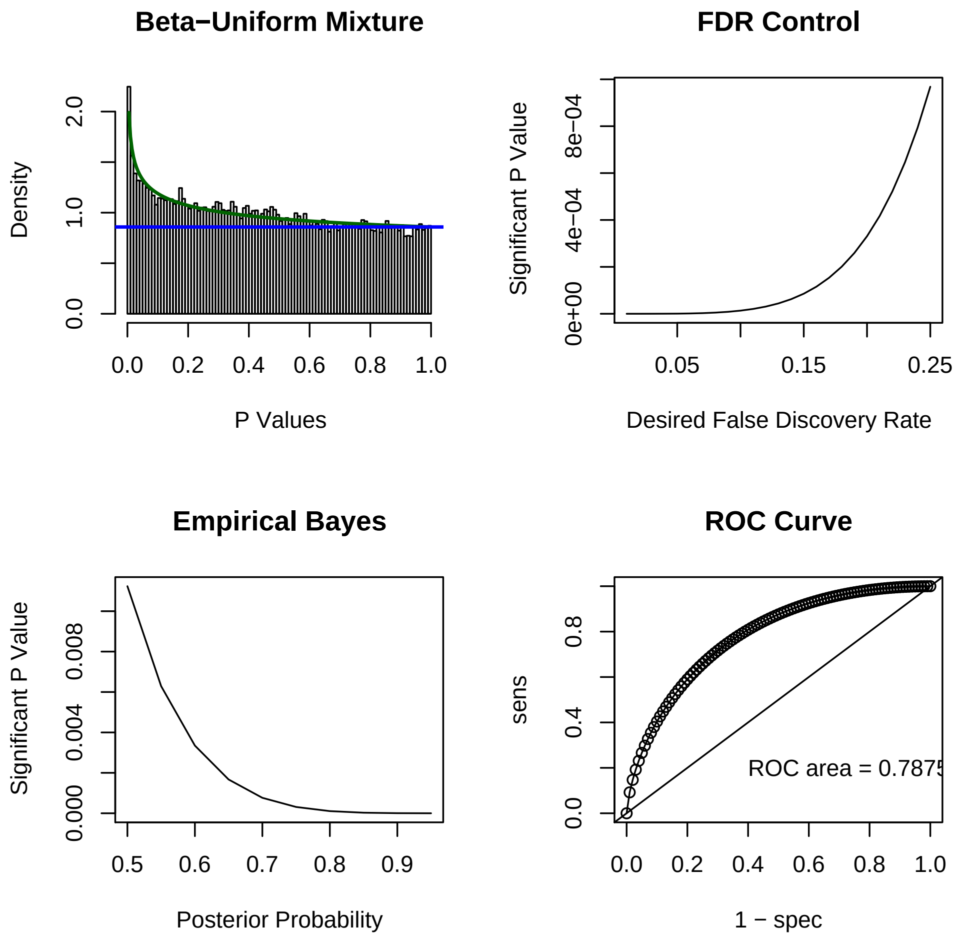


Figure 7: Results of the BUM analysis. Upper left panel is a histogram of p-values, with overlaid curves representing the beta and uniform components.

```
Row-by-row two-sample t-tests with 54675 rows
Positive sign indicates an increase in class: Center

Call: MultiTtest(data = x.mas, classes = "Location")

T-statistics:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-42.08000  -0.91800  -0.10470  -0.06304   0.74770  31.45000

P-values:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
1.906e-06 2.106e-01 4.496e-01 4.666e-01 7.165e-01 1.000e+00
```

The summary shows that there are some extremely large t-statistics. It also points out that positive t-statistics indicate overexpression of a gene in the center of the tumor, so negative t-statistics indicate higher expression in the periphery of the tumor.

To adjust for multiple testing, we use a beta-uniform mixture (BUM) model to estimate the false discovery rate (FDR). A plot of the fitted model shows that there are definitely some differentially expressed genes, but also suggests that the false discovery rates are likely to be fairly high even at very small p-value cutoffs (Figure 8).

```
> mas5.bum <- Bum(mas5.mtt@p.values)
> mas5.sel <- selectSignificant(mas5.bum, alpha = 0.35, by = "FDR")
> countSignificant(mas5.bum, alpha = 0.35, by = "FDR")

[1] 207
```

4.3 Differential Expression using Median-Normalized MAS5

The `MultiTtest` function performs gene-by-gene two-sample t-tests. The second argument is a character string indicating which factor in the phenoData object should be used for the comparison.

```
> mas5med.mtt <- MultiTtest(x.mas5med, "Location")
> summary(mas5med.mtt)

Row-by-row two-sample t-tests with 54675 rows
Positive sign indicates an increase in class: Center

Call: MultiTtest(data = x.mas5med, classes = "Location")

T-statistics:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-30.78000  -0.73230   0.08315   0.10800   0.89680  43.83000

P-values:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
1.620e-06 2.225e-01 4.591e-01 4.733e-01 7.194e-01 1.000e+00
```

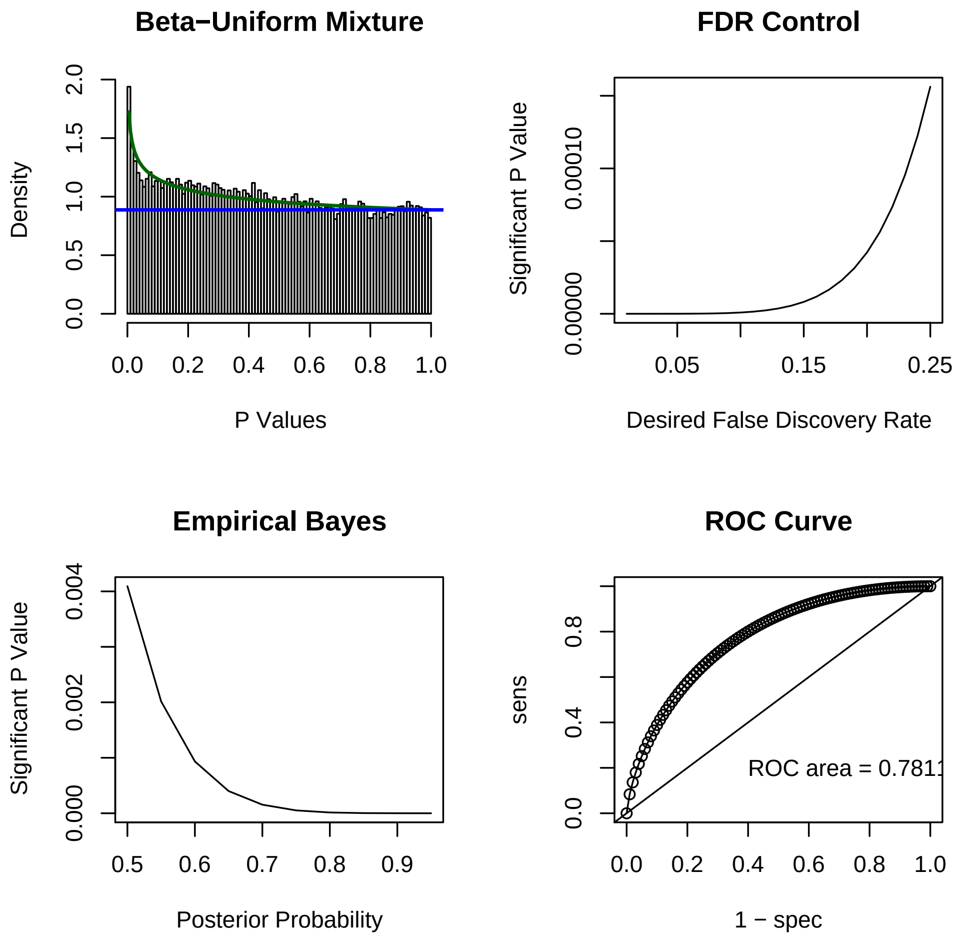


Figure 8: Results of the BUM analysis. Upper left panel is a histogram of p-values, with overlaid curves representing the beta and uniform components.

The summary shows that there are some extremely large t-statistics. It also points out that positive t-statistics indicate overexpression of a gene in the center of the tumor, so negative t-statistics indicate higher expression in the periphery of the tumor.

To adjust for multiple testing, we use a beta-uniform mixture (BUM) model to estimate the false discovery rate (FDR). A plot of the fitted model shows that there are definitely some differentially expressed genes, but also suggests that the false discovery rates are likely to be fairly high even at very small p-value cutoffs (Figure 9).

```
> mas5med.bum <- Bum(mas5med.mtt@p.values)
> mas5med.sel <- selectSignificant(mas5med.bum, alpha = 0.5, by = "FDR")
> countSignificant(mas5med.bum, alpha = 0.5, by = "FDR")

[1] 168
```

4.4 Comparing the Gene Lists

Now we want to see how similar (or different) are the lists of differentially expressed genes from different processing methods. We start by simply counting the overlap:

```
> temp <- as.matrix(data.frame(RMA = rma.sel, MAS5 = mas5.sel,
+     MAS5M = mas5med.sel)) * 1
> overlap <- t(temp) %*% temp

> xtable(overlap, digits = rep(0, 4))
```

	RMA	MAS5	MAS5M
RMA	198	42	29
MAS5	42	207	51
MAS5M	29	51	168

The overlap is not nearly as great as we might have liked, being on the order of 20%. Even when we compare MAS5 data to median-normalized MAS5 data, the overlap is relatively small.

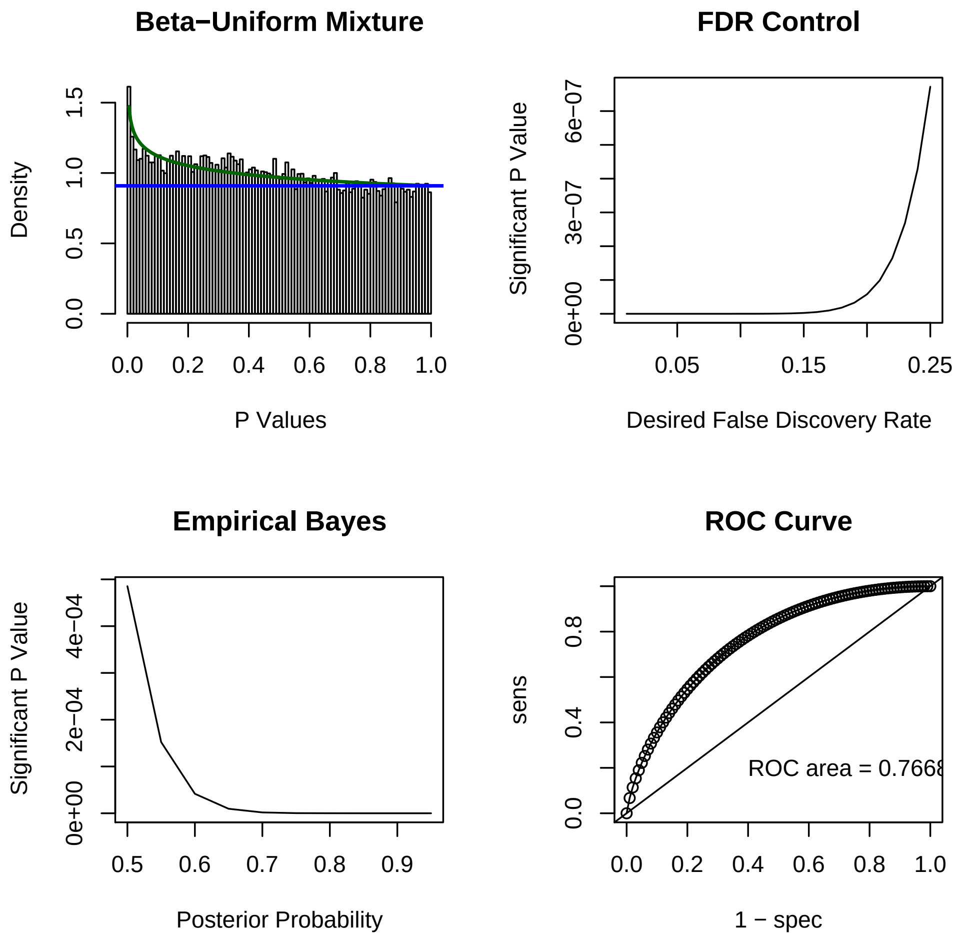


Figure 9: Results of the BUM analysis. Upper left panel is a histogram of p-values, with overlaid curves representing the beta and uniform components.