Analysis of Murine Palate Two-Color miRNA Microarray Data

Guy N. Brock[1], Partha Mukhopadhyay[2], Vasyl Pihur[1],
Cynthia Webb[2], Robert M. Greene[2], and M. Michele Pisano[2]

[1]Department of Bioinformatics and Biostatistics
[2]Department of Molecular Cellular and Craniofacial Biology
University of Louisville

May 3, 2016

Contents

Abstract

This vignette presents an analysis of a typical two-color miRNA microarray experiment. Covered topics include visualization, normalization, quality checking, differential expression, cluster analysis, miRNA target identification, and gene set enrichment analysis. Many of these tools carry-over from the analysis of mRNA microarrays, but with some notable differences that require special attention.

1 Introduction

In this vignette, we present the analysis of a two-color miRNA microarray experiment. The *MmPalateMiRNA* package is a compendium [**?**, **?**], which encapsulates the primary data, supporting software and statistical analysis, and document text. The data were obtained from mouse embryos during the development of the secondary palate [**?**], and are freely available as part of the compendium package. The analysis presented here follows closely to what was presented in [**?**], though with a few differences. Although the analysis is specific to the Miltenyi Biotech miRXplore platform [1], the general steps outlined here can easily be extended to other platforms as well.

Example code is provided for the complete analysis including preprocessing of arrays, normalization, identification of differentially expressed miRNAs, clustering, miRNA target identification, and gene-set enrichment analysis. Aspects of miRNA analysis which require special attention are highlighted. In particular, miRNA arrays typically have far fewer genes that are spotted on the array compared to mRNA arrays, and require careful consideration of the assumptions behind array pre-processing methods prior to their application. Several recent publications have compared various normalization methods for microRNA microarray data [**?**, **?**, **?**], while others have developed novel methods specifically for miRNA data [**?**, **?**, **?**, **?**]. Though certain methods were found to outperform others in each case, in general there is still no consensus on the best normalization method. In this vignette we illustrate how to assess whether a normalization method is appropriate for a given data set, using the diagnostic plots discussed in [**?**]. A second unique aspect of miRNA analysis relative to mRNA analysis is that differentially expressed miRNAs are subsequently evaluated for potential gene targets that are regulated by the miRNAs. A number of databases can be used for this purpopse, and many of these have been ported to R in the form of Bioconductor packages. It is these putative regulatory targets that are typically evaluated for biological and molecular functionality, e.g. by gene set enrichment analysis.

2 miRNA Palate Data

The microRNA microarray data in this compendium were obtained as previously described in [**?**], and the data are publicly available from GEO DataSets [2], (accession number GPL10179). Briefly, mouse embryonic tissue was obtained on gestational days (GD) 12, 13, and 14, which represents the critical period of palate development in the mouse. Total RNA (containing

[1] Miltenyi Biotec: products & services for biomedical research, `http://www.miltenyibiotec.com/en/default.aspx`

[2] GEO DataSets home, `http://www.ncbi.nlm.nih.gov/gds/`

miRNAs) was isolated using standard RNA extraction protocols. RNA samples (1 μg) isolated from mouse embryonic orofacial tissues (GD-12 - GD-14) as well as the miRXplore Universal Reference (control) were fluorescently labeled with Hy5 (red) or Hy3 (green), respectively, and hybridized to miRXplore Microarrays (Miltenyi Biotec) using the a-Hyb Hybridization Station (Miltenyi Biotec). For each gestational day, three distinct pools of RNA were independently processed and applied to microarray chips. Probes for a total of 1336 mature miRNAs (from human, mouse, rat and virus), including positive control and calibration probes, were spotted in quadruplicate on each microarray. Each array included probes for 588 murine miRNAs. The miRXplore Universal Reference (UR) controls, provided by Miltenyi, represent a defined pool of synthetic miRNAs for comparison of multiple samples. Fluorescence signals of the hybridized miRXplore Microarrays were detected using a laser scanner from Agilent Technologies. Mean and median signal and local background intensities for the Hy3 and Hy5 channels were obtained for each probe on each of the nine microarray images using the ImaGene software (BioDiscovery, Inc., El Segundo, CA).

The raw data are available in the package as `PalateData`, and are loaded below. Additional meta-information concerning the type of probe (`probe.type`) and name of the miRNA (`Name` and `Name.stem`) are available in the `PalateData$genes` data frame. See the help page for details. The data are in the format of an `RGList`, and requires the *limma* package.

```
R> library("MmPalateMiRNA")
R> data(PalateData)
```

3 Preprocessing

3.1 Outlying Arrays

Sarkar *et al.* [**?**] describe several diagnostic plots for miRNA data that can be used to evaluate the need and effectiveness of normalization procedures. These plots can also serve as aids to determine outlying arrays and batch effects. One such plot is the kernel density estimate for each array, for different types of probes. Figure **??** plots the density estimates of the $\log_2$ intensity values in the control channel for the unnormalized data, separated into "MMU miRNAs", "Other miRNAs", and "Control" probes (other probes were non-informative). The plot requires use of the *lattice* package, and the *MmPalateMiRNA* package contains methods to produce plots for `RGList` objects based on the generic functions in *lattice*.

```
R> ######################################################################
R> ## Densities of log2 intensities in reference (green) channel
R> ## organize according to different types of probes
R> ## different lines correspond to different arrays
R> ######################################################################
R>
R> ## separate into mouse, non-mouse, blank, control probes
R> ## panels = probe type AND normalization type
R> ## groups = arrays
```

```
R> ## y = density
R> ## x = log2 values
R>
R> ## Use ~ x1 + x2 | y fomula for multiple x's ..
R> ## Done within the S4 method for class 'RGList' for densityplot
R> ## Code so that each day is different color, but each replicate is different line type
R>
R> #library("lattice")
R>
R> res <- densityplot(PalateData, channel="G", group="probe.type",
                     subset = c("Other miRNAs",   "MMU miRNAs", "Control"),
                     col=rep(1:3, each=3), lty=rep(1:3, 3),
                     key = list(lines=list(col=rep(1:3, each=3), lty=rep(1:3, 3)),
                       text=list(colnames(PalateData)), columns=3))
R> print(res)
R>
R> ## looks like 3 outlying arrays (021, 033, 029)
R> ## 12-1, 14-3, 13-2 ...
R> ## 12-1, 12-2, 12-3, 13-1, 13-2, 13-3, 14-1, 14-2, 14-3
R> ## 021,  022,  023,  024,  033,  034,  035,  036,  029
R>
```

Figure **??** indicates three possible outlying arrays, GD 12-1, 13-2, and 14-3. A second figure (Figure **??**) can be constructed based on the pairwise "distance" between arrays, as measured by the median of the absolute differences in $\log_2$ expression for miRNAs in the green channel [**?**]. The plot is created using the `levelplot` method for `RGList` objects included in the package. Here we separate the plots according to the type of probe, and the arrays are reordered so that the outlying arrays are grouped together. The three arrays are clearly outliers based on the control probes, but to a lesser extent based on the other types of probes.

```
R> ######################################################################
R> ## Plot of pairwise "distance" between arrays
R> ## distance defined as median absolute difference between arrays for MMU miRNAs
R> ## use log2 of green channel
R> ######################################################################
R>
R> ## levelplot using S4 method for class 'RGList'
R> ## reorder so that outlying arrays are grouped together
R> res <- levelplot(PalateData[, c(1,5,9,2:4,6:8)],
                   channel="G", group="probe.type",
                   subset=c("MMU miRNAs", "Other miRNAs", "Control", "Empty"),
                   scales = list(rot=c(45, 45)))
R> print(res)
R>
```

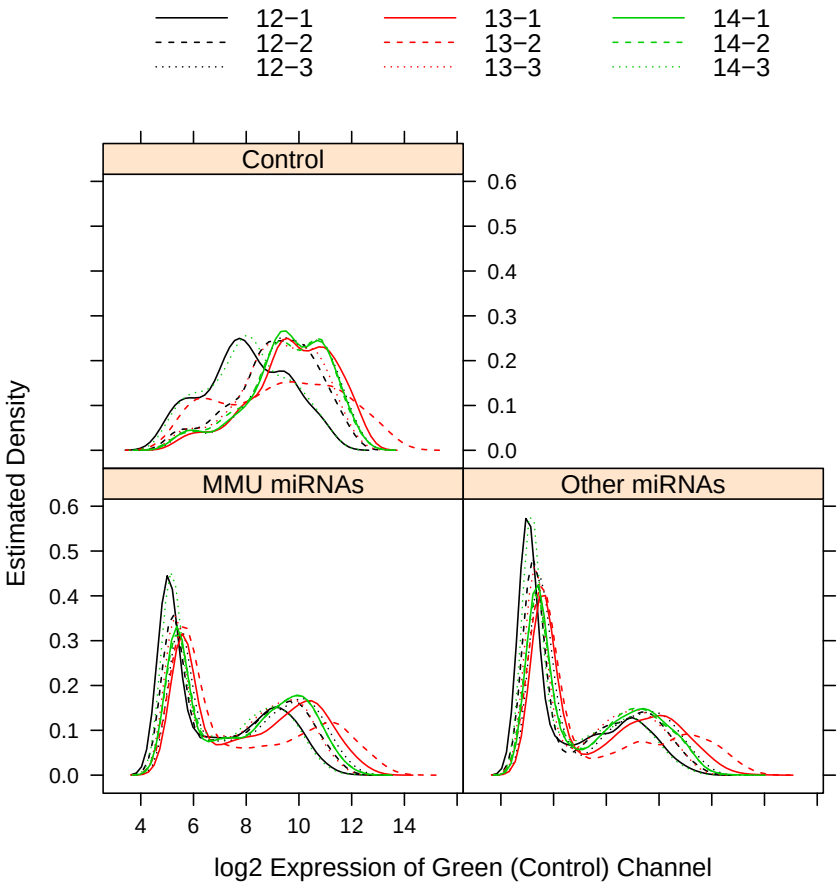

Figure 1: Estimated density of reference channel in unnormalized data, for various types of probes.

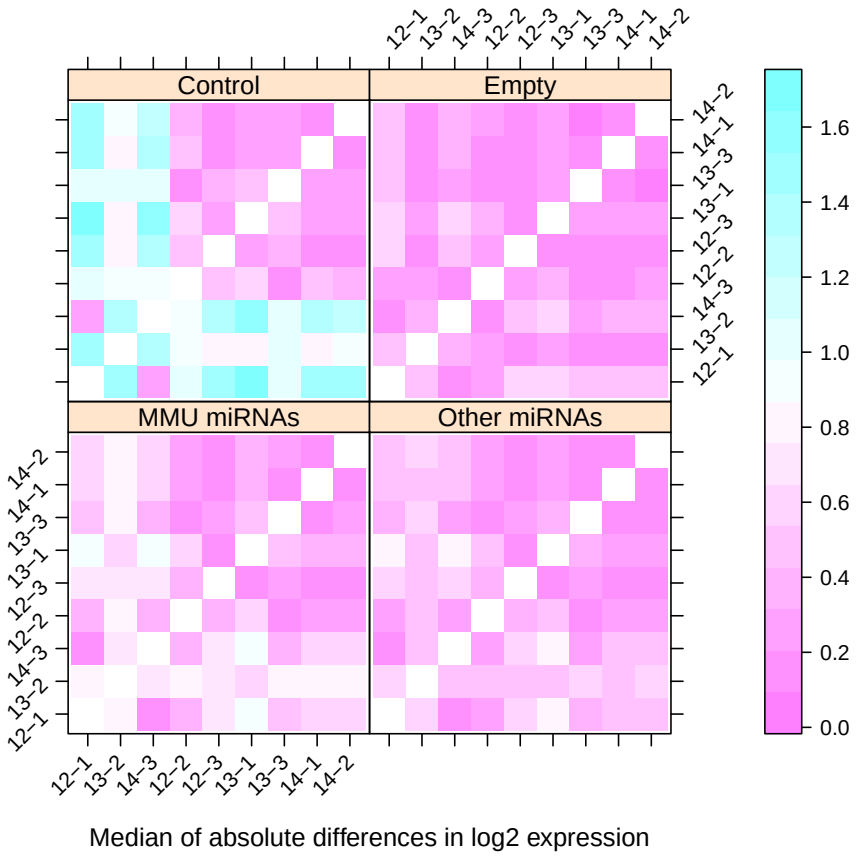

Figure 2: Distance between arrays, as measured by the median absolute distance of reference channel intensity measurements, separate by type of probe.

Figures **??** and **??** demonstrate the potential need for normalization or removal of several of the arrays. In Section **??**, we will evaluate the effectiveness of several normalization methods in correcting these systematic differences between the arrays.

3.2 Outlying Values

In addition to checking for outlying arrays, it is important to check for outlying values on individual probes. To accomplish this, we evlauted for each probe whether there were any extreme values (greater than 2.665 standard deviations above the mean). The `checkOutliers` function checks this for each of the red and green channels in an `RGList` object, and returns the indices of array probes with extreme values.

```
R> outliers <- checkOutliers(PalateData)
```

The probes with outlying arrays can be visualized using boxplots, e.g. using the code below.

```
R> boxplot(as.data.frame(t(PalateData$R[outliers$Rout,])))
```

The figure is omitted but clearly shows that the identified outlying values are nearly two orders of magnitude above the rest of the intensity values. Rather than omitting these values, we exploit the replicated design of the arrays and substitute the mean of the other replicates on the array for the extreme values (the `fixOutliers` function).

```
R> PalateData$R <- fixOutliers(PalateData$R, outliers$Rout, PalateData$genes$Gene)
R> PalateData$G <- fixOutliers(PalateData$G, outliers$Gout, PalateData$genes$Gene)
R>
```

3.3 Missing Values

In addition to checking for outlying values, we also check for any missing values in the two channels (the `checkMVs` function). Here, we only find two probes on the array with missing values in the background channels, so we again impute these values using the means of the backgrounds from the other three replicates on the chip (the `fixMVs` function).

```
R> mvs <- checkMVs(PalateData)
R> PalateData$Rb <- fixMVs(PalateData$Rb, mvs$Rb.na, PalateData$genes$Gene)
R> PalateData$Gb <- fixMVs(PalateData$Gb, mvs$Gb.na, PalateData$genes$Gene)
R> ## Three spots with missing backgrounds
R> ## 39875 (1 Rb and 1 Gb)
R> ## 38232 (2 Rb and 2 Gb)
R> ## 40139 (3 Rb and 3 Gb)
R>
```

3.4 Filtering Probes

Prior to running the normalization methods, we filter the probes and keep only those which correspond to miRNAs and calibration probes. Additionally, probes that are not sufficiently above the background intensity level may be unreliable and represent noise that can interfere with subsequent analysis, including normalization [**?**]. Prefiltering also reduces the number of statistical comparisons being performed and improves overall power [**?**]. Here, we filter probes whose foreground intensity values are below 1.1 times their background intensity level. To allow for probes which may be expressed for a particular experimental condition (here, gestational day), we keep all probes which have at least 3 samples above the filtering threshold. Lastly, only those genes with all four replicates passing the filtering step are retained. After all pre-processing steps, a total of 956 probes, corresponding to 175 mouse miRNAs, 42 other miRNAs, and 22 calibration probes each replicated 4 times, remains.

```
R> reducedSet <- filterArray(PalateData, keep=c("MIR", "LET", "POSCON", "CALIB"),
                             frac=1.1, number=3, reps=4)
R>
```

4 Normalization

Based on the literature [**?**, **?**, **?**, **?**], we evaluated several normalization procedures on the filtered data, including none, median, loess, quantile, VSN, and spike-in VSN. The *limma* package includes various options for both within (`normalizeWithinArrays`) and between (`normalizeBetweenArrays`) array normalization, and the *vsn* package has functions for performing VSN and spike-in VSN. In all cases, a simple background correction was performed by subtracting background from the foreground intensities.

```
R> ######################################################################
R> ## Evaluate different normalization procedures
R> ## 1. none
R> ## 2. median
R> ## 3. loess
R> ## 4. quantile
R> ## 5. VSN
R> ## 6. spike-in VSN
R> ######################################################################
R>
R> ## should just require this??
R> library("vsn")
R> ## 1. None
R> ndata.none <- normalizeWithinArrays(reducedSet, method="none")
R> ## 2. Median
R> ndata.median <- normalizeWithinArrays(reducedSet, method="median")
R> ## 3. Loess
R> ndata.loess <- normalizeWithinArrays(reducedSet, method="loess")
```

```
R> ## 4. Quantile
R> ndata.quantile <- normalizeBetweenArrays(reducedSet, method="quantile")
R> ## 5. VSN
R> ndata.vsn.limma <- normalizeVSN(reducedSet)
R> ## NOTE: can also get the above with the following code:
R> ## ndata.vsn <- justvsn(reducedSet, backgroundsubtract=TRUE)
R> ## However, 'ndata.vsn' is an NChannelSet while
R> ##   'ndata.vsn.limma' is an MAList
R>
R> ## 6. Spike-in VSN
R> idx.control <- which(reducedSet$genes$probe.type=="Control")
R> spikein.fit <- vsn2(reducedSet[idx.control,], lts.quantile=1, backgroundsubtract=TRUE)
R> ndata.spikein.vsn <- predict(spikein.fit, newdata=reducedSet)
R>
```

4.1 Diagnostic Plots

Several diagnostic plots can be used to contrast the effectiveness of each normalization procedure, as suggested by [**?**]. The *MmPalateMiRNA* package contains several methods to produce these plots for lists of class `MAList` or `NChannelSet` objects, based on functions in the *lattice* package. Figure **??**, rows one through five, plots the intensity distribution for the reference channels after each of the normalization procedures (use of the `useOuterStrips` function requires the *latticeExtra* package). The quantile normalization procedure is clearly the most successful in removing the intensity bias that was apparent for three of the arrays (12-1, 13-2, and 14-3), while loess and median normalization appear to be the least successful. Notably, normalization based on the spike-in probes was unsuccessful, perhaps since these probes were shifted differently in the reference channel relative to the other probe types.

```
R> ######################################################################
R> ## 1. Evaluation using Sarkar et al. 2009 density plots
R> ##    Use RG.MA to convert back to RG values for MALists
R> ##    Use density plot - S4 method for class 'list'
R> ######################################################################
R>
R> ndata.all <- list(ndata.none, ndata.median, ndata.loess,
                    ndata.quantile, ndata.vsn.limma,
                    ndata.spikein.vsn)
R> names(ndata.all) <- c("None", "Median", "Loess", "Quantile", "VSN", "Spike-in VSN")
R> dplot <- densityplot(ndata.all, channel="G", group="probe.type",
                        subset = c("Other miRNAs",   "MMU miRNAs", "Control"),
                        col=rep(1:3, each=3), lty=rep(1:3, 3),
                        par.strip.text=list(cex=0.75),
                        key = list(lines=list(col=rep(1:3, each=3), lty=rep(1:3, 3)),
                          text=list(colnames(ndata.none)), columns=3))
R> if (require("latticeExtra")) {
```

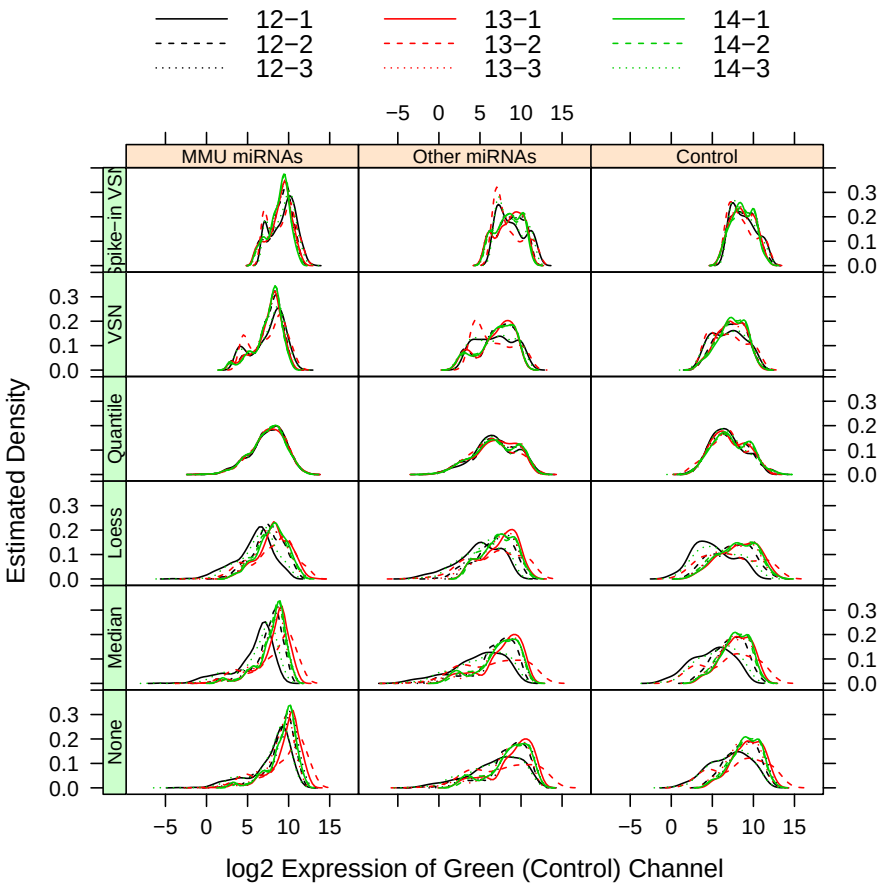

Figure 3: Estimated density of reference channel before ("None") and after normalization by various normalization procedures, subsetted by type of probe.

```
   dplot <- useOuterStrips(dplot)
 }
R> plot(dplot)
```

An additional plot based on the median absolute difference between probes in the reference channel can be used to compare relative success of the normalization procedures in removing the array effect (Figure **??**). Here again, quantile normalization appears to be the best, while loess and median normalization are the least effective.

```
R> ######################################################################
R> ## 2. Sarkar 'Distance' Plot
R> ##    Green channel, all probes
R> ######################################################################
R>
R> ## levelplot using S4 method for class 'list'
R> res <- levelplot(ndata.all, channel="G", order=c(1,5,9,2:4,6:8),
```

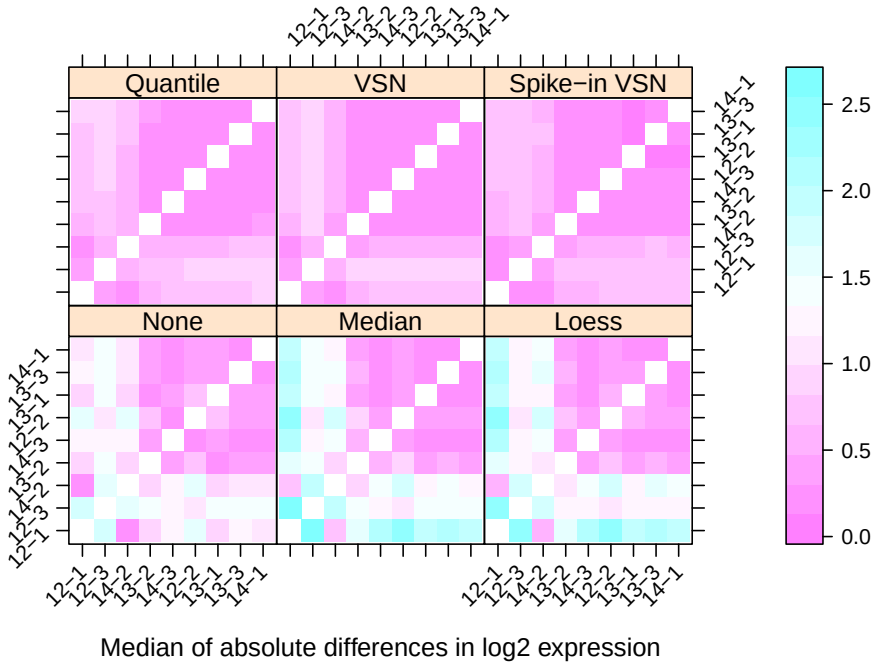

Figure 4: Distance between arrays, as measured by the median absolute distance of reference channel intensity measurements, after application of several normalization procedures.

```
                    scales = list(rot=c(45, 45)))
R> print(res)
R>
R>
```

To investigate the effect of the normalization procedure on the experimental channel, plots of the spread (median absolute deviation) versus the location (median) of all probes can be used. Plots of this type can be produced using the `MADvsMedianPlot` function in the package. Probes of different types are highlighted, with particular focus on the spike-in probes, which should have low variability across all the arrays. In Figure **??**, spike-in VSN has the lowest variability among the spike-in probes, compared to the other normalization methods. However, spike-in VSN has also dramatically decreased the variation among *all* the probes in the experimental channel, making the normalization procedure questionable in this case. Quantile normalization has resulted in large variations for some of the probes with lower intensity values.

```
R> ######################################################################
R> ## 3. Spread (MAD) vs. location (median) of all probes,
R> ##     highlight spike-ins
R> ######################################################################
R>
R> ## Previously did MAD vs median of expression-ratios (M values) across all arrays
R> ## Now doing MAD vs median for RED channel ...
R> ## Either plot may be interesting ...
R>
R> ## Now using S4 method MADvsMedianPlot ...
R> res <- MADvsMedianPlot(ndata.all, channel="R", group="probe.type",
                  subset=c("MMU miRNAs", "Other miRNAs", "Control"))
R> print(res)
R> ## Here, maybe quantile looks a little suspect??
R> ## VSN, quantile, and loess are close
R> ## spike-in VSN very little variation and is suspect
R>
```

Plots of the $\log_2$ expression ratios (M values) versus the mean $\log_2$ expression values (A values) for each probe can be used to evaluate whether their is a bias associated with overall intenstity level for each array. This so-called "MA" plot is illustrated in Figure **??** for quantile normalization. MA plots for the other normalization methods are not shown, though code to produce the plots is available in the R script for the vignette. Quantile normalization has removed any association between the M and A values, while for VSN normalization there is still a trend which is similar to the unnormalized data The MA plot for spike-in VSN shows a dramatic effect on the intensity ratios.

```
R> ######################################################################
R> ## MA Plots for
R> ## 1. quantile normalization
R> ## 2. VSN
R> ## 3. unnormalized data
R> ## 4. spike-in VSN
R> ######################################################################
R>
R> ## Quantile
R> res <- MAplot(ndata.quantile)
R> print(res)
R>
```

As a final evaluation, we inspection heatmaps along with hierarchical clustering of the arrays. Figure **??** displays the heatmap after quantile normalization, and reveals that the previously identified outlying arrays (samples 12-1, 13-2, and 14-3) still do not cluster with the other replicates for that day.

```
R> ######################################################################
R> ## Heatmaps
```

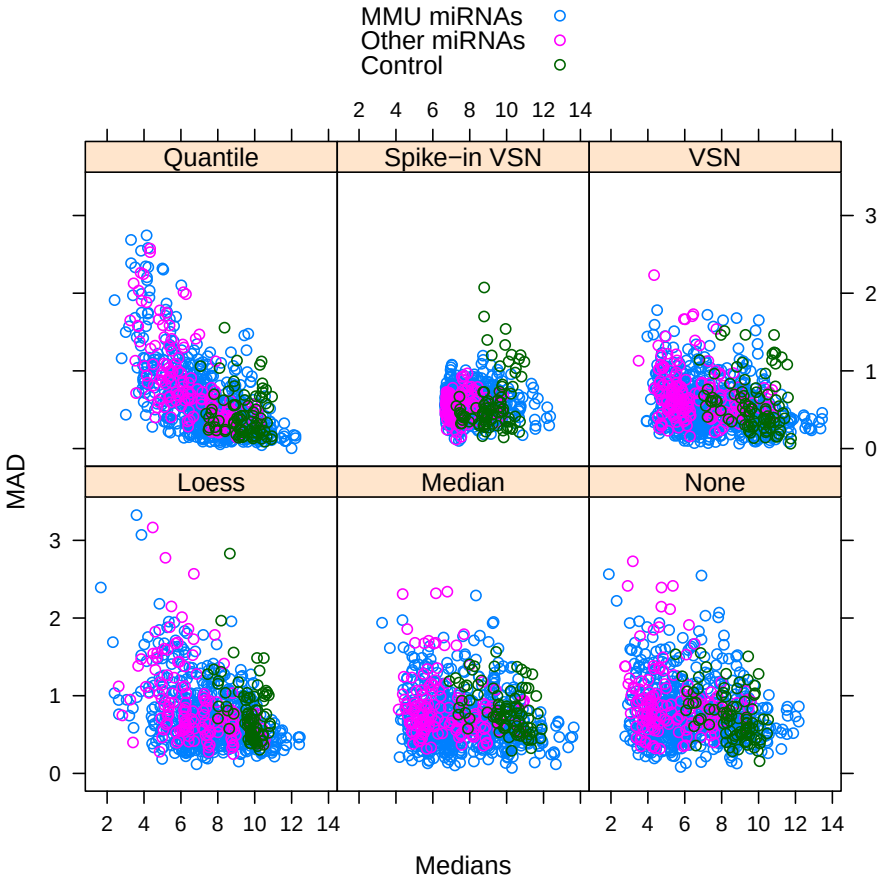

Figure 5: Spread, as measured by the median absolute deviation (MAD), versus median expression for each probe in the experimental channel after application of several normalization procedures. Points are color-coded by type of probe.

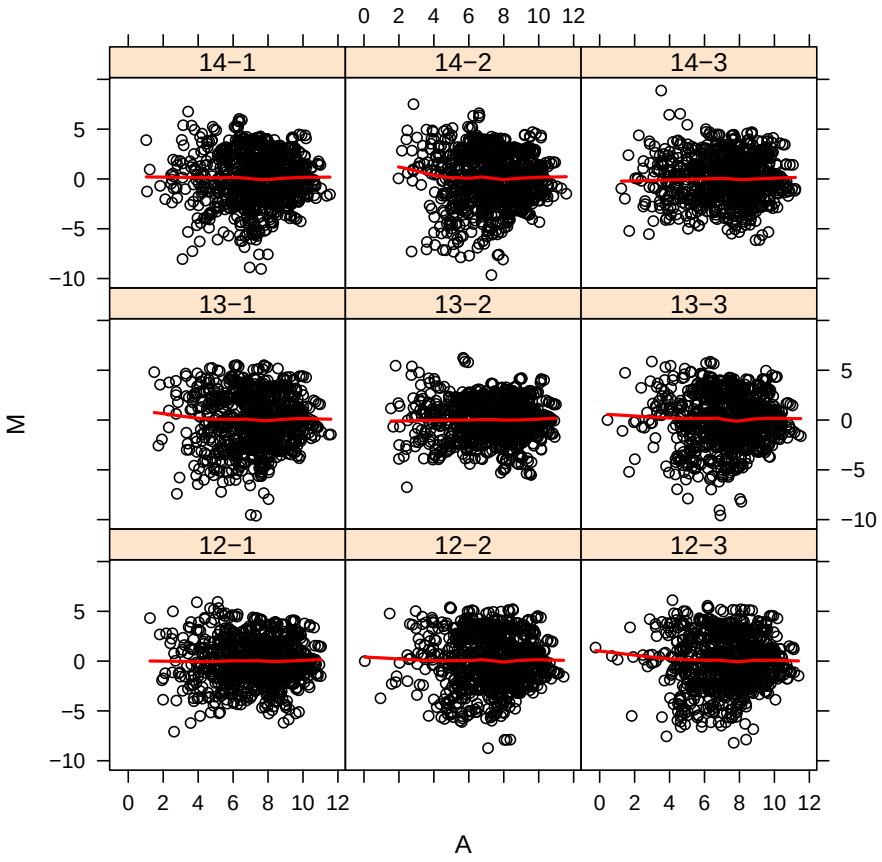

Figure 6: Log_2 intensity ratios plotted against average $\log_2$ intensity values for each probe, for each array after quantile normalization. Red lines are loess smoothed regression lines for each M versus A comparison.

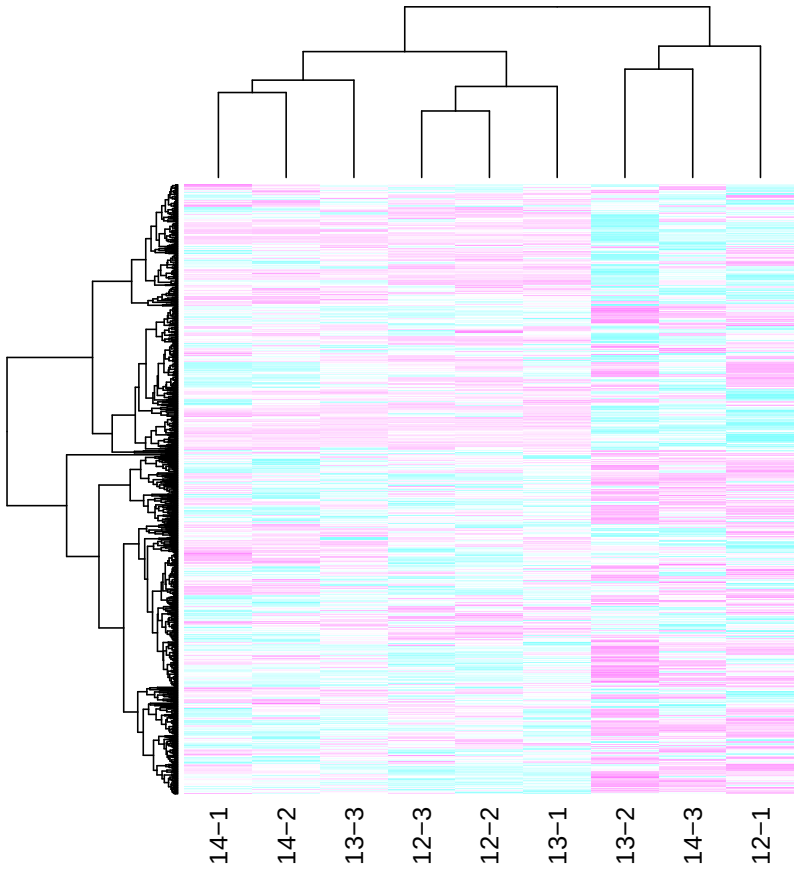

Figure 7: Heatmap of $\log_2$ intensity ratios after quantile normalization. Arrays and probes are clustered by hierarchical clustering. Arrays 12-1, 13-2, and 14-3 do not cluster with the other replicates for the corresponding gestational day.

```
R> ######################################################################
R> ## Clusters both rows and columns
R> heatmap(ndata.quantile$M, col=cm.colors(256), labRow=FALSE)
R>
```

Table **??** gives the correlations between each pair of arrays, based on the $\log_2$ intensity ratios. Since the other two replicates for each day were highly correlated ($r \geq 0.95$), we decided to use only those two replicates from each day for subsequent statistical analysis. Normalization was redone omitting the arrays 12-1, 13-2, and 14-3, using quantile normalization.

```
R> ######################################################################
R> ## Redo, omiting 021, 033, and 029 (12-1, 13-2, 14-3)
R> ## keep two replicates for each day
R> ## Removal before normalization
```

	12-1	12-2	12-3	13-1	13-2	13-3	14-1	14-2	14-3
12-1	1.00	0.86	0.86	0.85	0.86	0.84	0.83	0.84	0.89
12-2	0.86	1.00	0.98	0.97	0.87	0.96	0.93	0.94	0.89
12-3	0.86	0.98	1.00	0.96	0.85	0.95	0.92	0.93	0.88
13-1	0.85	0.97	0.96	1.00	0.87	0.95	0.94	0.94	0.91
13-2	0.86	0.87	0.85	0.87	1.00	0.86	0.84	0.84	0.91
13-3	0.84	0.96	0.95	0.95	0.86	1.00	0.95	0.95	0.88
14-1	0.83	0.93	0.92	0.94	0.84	0.95	1.00	0.96	0.91
14-2	0.84	0.94	0.93	0.94	0.84	0.95	0.96	1.00	0.89
14-3	0.89	0.89	0.88	0.91	0.91	0.88	0.91	0.89	1.00

Table 1: Correlation between arrays after quantile normalization

```
R> ########################################################################
R>
R> ## Need to take out "MIRCONTROL" samples now as well
R> ## NOTE: Keeping these in improves correlation between samples on same day
R> ##       Remove prior to calculating whether gene is expressed and DE
R>
R> ########################################################################
R> ## 1. Quantile between, no within
R> ## 2. VSN
R> ########################################################################
R>
R> ## Background correction = 'subtract' by default (all methods)
R> omit <- which(colnames(reducedSet)%in%c("12-1", "13-2", "14-3"))
R> ndata <- normalizeBetweenArrays(reducedSet[,-omit], method="quantile")
```

4.2 Imputation

Sixteen probes from the six arrays exhibited negative intensities after the background procedure, and resulted in missing values for subsequent calculation of the $\log_2$ intensity ratios. A significant percentage of missing values can have a substantial impact on downstream analysis of array data [**?**], and in such cases choice of a imputation procedure should be carefully considered. Here, with a relatlively small percentage of missing values, the impact on data analysis will be relatively minimal, and we select the K-nearest neighbor imputation scheme [**?**] as a fast and effective approach (the `imputeKNN` function is included in the package).

```
R> ndata$M <- imputeKNN(as.matrix(ndata$M), dist="cor")
R> ndata$A <- imputeKNN(as.matrix(ndata$A), dist="cor")
```

5 Determining Differentially Expressed miRNAs

To test for differential expression of miRNAs between different gestational days (GD-12, 13, and 14), the *limma* package [**?**] was used. Use of the *limma* package requires the user

to create a design matrix, which defines the possible levels for each experimental factor, and is used to construct a model matrix and contrasts to test for differential expression between factor levels. The model matrix consists of indicator variables for the levels of each experimental factor in our design, which in our case corresponds to each of the gestational days.

```
R> ## Construct the design matrix
R> design <- data.frame(grp=c(1,1,2,2,3,3),rep=c(1,2,1,2,1,2))
R> design$grp <- factor(design$grp, labels=c("Day12", "Day13", "Day14"))
R> mmat <- model.matrix( ~ 0 + design$grp)
R> colnames(mmat) <- c("Day12", "Day13", "Day14")
```

The contrasts defined here estimate the differences in mean $\log_2$ expression between each gestational day. The `makeContrasts` function in *limma* will generate these for you.

```
R> contrast.matrix <- makeContrasts(Day13-Day12, Day14-Day12, Day14-Day13,
                                    levels=mmat)
```

Some advantages of using *limma* over other methods include the ability to incorporate probe quality weights and to handle duplicate probes for each miRNA on the chip via the `duplicateCorrelation` function. These advantages are particularly evident in small sample sizes, as in this experiment. To make use of the duplicated probes, we first order the normalized data so that replicated probes are adjacent to each other. The probe quality weights are incorporated in the calculation of the correlation matrix for the duplicated probes.

```
R> ## Order data by probes
R> idx <- order(ndata$genes$Gene)
R> ndata <- ndata[idx,]
R> idx.rm <- which(ndata$genes$probe.type=="Control")
R> ndata <- ndata[-idx.rm,]
R> ## compute correlations between same probes on each chip
R> corfit <- duplicateCorrelation(ndata, mmat, ndups=4,
                                  weights=ndata$weights)
R>
```

Next, the `lmFit` function is used to fit the hierarchical linear model, and the `contrasts.fit` function used to get contrast estimates. The `eBayes` function generates the moderated (empirical Bayesian) t-statistics corresponding to each of the contrast estimates.

```
R> fit <- lmFit(ndata, mmat, ndups=4,
                correlation=corfit$consensus)
R> fitc <- contrasts.fit(fit, contrast.matrix)
R> fitc <- eBayes(fitc)
```

The `topTable` function calculates and reports fold change, moderated t-statistics, unadjusted and adjusted p-values for the comparison of interest. Code below shows the calculation for the comparison between gestational days 13 and 12, and the results are given

in Table **??**. Results for comparisons between the other gestational days are omitted but code to calculate them is inclued in the R script for the vignette.

```
R> ## First contrast
R> top13v12 <- topTable(fitc, coef=1, number=nrow(ndata)/4, adjust="fdr",
                        sort.by="P")
R> top13v12$FC <- 2^(top13v12$logFC)
R> sig13v12 <- top13v12[top13v12$adj.P.Val<.05,]
R> colNames <- c("Name.stem", "probe.type", "FC", "t", "adj.P.Val")
R> res <- xtable(sig13v12[,colNames],
                 digits=c(0,0,0,2,2,3),
                 caption="Significantly differentially expressed miRNAs
                          for GD 13 versus 12",
                 label="tab:contrast13v12", caption.placement = "top")
R> print(res, include.rownames=FALSE)
```

Name.stem	probe.type	FC	t	adj.P.Val
LET-7B	MMU miRNAs	1.78	7.71	0.000
MIR-193A-3P	MMU miRNAs	2.94	6.84	0.000
LET-7C	MMU miRNAs	1.50	5.73	0.001
MIR-140-5P	MMU miRNAs	1.46	5.31	0.001
MIR-342	Other miRNAs	0.56	-5.17	0.001
MIR-31	MMU miRNAs	1.56	4.98	0.002
MIR-193B	MMU miRNAs	1.66	4.86	0.002
MIR-301	Other miRNAs	0.78	-4.44	0.005
MIR-20B	Other miRNAs	0.75	-4.36	0.006
MIR-543-3P	MMU miRNAs	0.69	-3.90	0.016
MIR-342-3P	MMU miRNAs	0.58	-3.82	0.016
MIR-301B	Other miRNAs	0.71	-3.82	0.016
MIR-22	MMU miRNAs	1.34	3.78	0.016
MIR-152	MMU miRNAs	1.25	3.74	0.016
LET-7I	MMU miRNAs	1.35	3.74	0.016
MIR-298	MMU miRNAs	0.77	-3.44	0.030
MIR-148A	MMU miRNAs	1.34	3.41	0.031
MIR-210	MMU miRNAs	1.33	3.39	0.031
MIR-422A	Other miRNAs	1.67	3.34	0.033
MIR-23A	MMU miRNAs	1.29	3.32	0.033
MIR-20A	MMU miRNAs	0.79	-3.29	0.033
MIR-347	Other miRNAs	1.19	3.18	0.042

Table 2: Significantly differentially expressed miRNAs for GD 13 versus 12

A nice summary of the results for the comparisons between gestational days is a Venn diagram, which gives the number of up- and down-regulated genes for each comparison, along with the number in the intersection of these sets (Figure **??**).

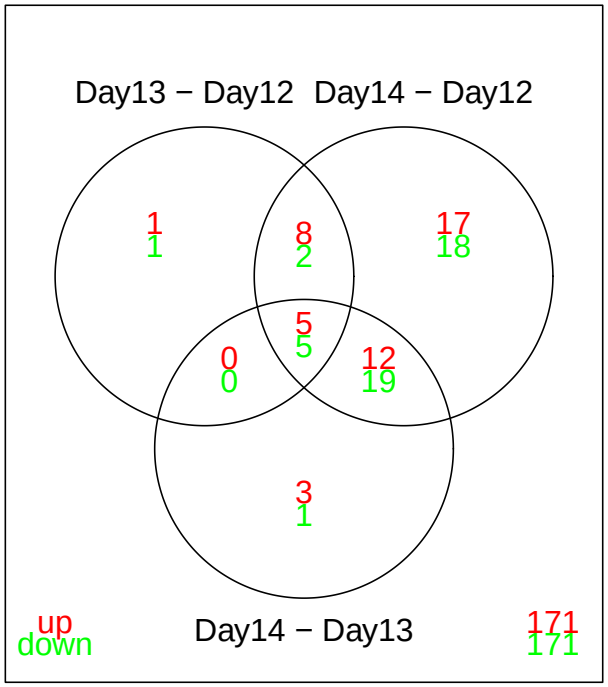

Figure 8: Venn diagram illustrating the number of up- and down-regulated genes for each comparison between gestational days, along with the number in the intersection of these sets.

```
R> res <- decideTests(fitc)
R> vennDiagram(res, include=c("up","down"),
               counts.col=c("red","green"), cex=1.25)
```

Although we have focused on the calculation of test statistics corresponding to pairwise comparisons between gestational days, it is easy to obtain estimates for other contrasts of interests between the experimental conditions. For example, the the `contr.poly` function will provide contrasts to test for linear and quadratic trends, and the `contr.helmert` function gives the Helmert contrasts. Additionally, the analysis of variance (ANOVA) F-statistics for testing for differential expression between all three gestational days are in `fitc$F`, with corresponding p-values in `fitc$F.p.value`.

To illustrate, we first calculate adjsuted p-values for the F-statistics in `fitc$F`, using the *multtest* package.

```
R> if (require("multtest")) {
   rawp <- fitc$F.p.value
   BH <-  mt.rawp2adjp(rawp, proc = "BH")
   sum(BH$adjp[,"BH"] < 0.05)
 }
```

```
[1] 79
```

The same statistics can be obtained by combining two orthogonal contrasts, e.g. using the `contr.helmert` function.

```
R> ## Helmert contrasts, from contr.helmert ...
R> contr.helmert(3)
```

```
  [,1] [,2]
1   -1   -1
2    1   -1
3    0    2
```

```
R> contrast.helmert <- makeContrasts(Day13-Day12, Day14-0.5*Day12-0.5*Day13,
                                     levels=mmat)
R> fitc.helmert <- contrasts.fit(fit, contrast.helmert)
R> fitc.helmert <- eBayes(fitc.helmert)
R> Fstats <- topTable(fitc.helmert, coef=c(1, 2), number=nrow(ndata)/4, adjust="fdr",
                      sort.by="F")
R> sum(Fstats$adj.P.Val < 0.05)
```

```
[1] 79
```

Next, the miRNAs with significant F-statistics (adjusted $p < 0.05$) are identified for follow-up examination, e.g. by clustering. The duplicates are averaged prior to further analysis.

```
R> ## average probes for plotting
R> avedata <- avedups(ndata, ndups=4, spacing=1)
R> sigFgenes <- Fstats$Gene.ID[which(Fstats$adj.P.Val < 0.05)]
R> ## ssgenes <- unique(c(sig13v12$Gene, sig14v12$Gene, sig14v13$Gene))
R> ## sum(sigFgenes%in%ssgenes) ## all 79 show up in individual lists
R> mat <- as.matrix(avedata[match(sigFgenes, avedata$genes$Gene), ])
R> colnames(mat) <- c("GD-12-1", "GD-12-2","GD-13-1", "GD-13-2",
                      "GD-14-1", "GD-14-2")
R> rownames(mat) <- sigFgenes
```

6 Clustering Expression Profiles

After identifying the differentially expressed miRNAs, clustering analysis can be performed to group genes with similar trends over time. A common difficultly is deciding which

clustering algorithm to use, and how many clusters to create. Cluster validation measures, as contained in the R package *clValid* [**?**], can help in this regard. Below, the `clValid` function is used to evaluate hierarchical clustering, SOTA, DIANA, and K-means clustering algorithms, for a range of one to six clusters in each case. The expression values for each day are averaged over the two replicates prior to clustering (object `aveExpr`). The internal validation measures (connectivity, Dunn Index, and Silhouette Width) are used with a correlation metric. A summary of the result indicates that hierarchical clustering with six clusters provides the optimal connectivity and Dunn Index measures, while DIANA with six clusters gives the optimal Silhouette Width.

```
R> if (require("clValid")) {
   ## This creates averages over replicates for each day
   aveExpr <- t(apply(mat, 1, function(x) tapply(x, c(1,1,2,2,3,3), mean)))
   clRes <- clValid(aveExpr, 6, clMethod=c("hierarchical", "diana", "sota","kmeans"),
                    validation=c("internal"), metric="correlation")
   summary(clRes)
 }

Clustering Methods:
 hierarchical diana sota kmeans

Cluster sizes:
 6

Validation Measures:
                                    6

hierarchical Connectivity   18.2833
             Dunn            0.0027
             Silhouette      0.7607
diana        Connectivity   21.0147
             Dunn            0.0033
             Silhouette      0.7708
sota         Connectivity  182.8651
             Dunn            0.0000
             Silhouette     -0.7568
kmeans       Connectivity  178.8687
             Dunn            0.0000
             Silhouette     -0.2833

Optimal Scores:

             Score   Method       Clusters
Connectivity 18.2833 hierarchical 6
Dunn          0.0033 diana        6
Silhouette    0.7708 diana        6
```

```
R>
```

The results from hierarchical clustering with six clusters was subsequently selected for visually displaying the data, using the `clustPlot` function available in this package. The expression values for each miRNA are scaled to mean zero and standard deviation one for ease of visualization. The display is given in Figure **??**. The two predominant clusters are cluster one and cluster two, which correspond to those miRNAs which exhibit a linear upward and downward trend over the time course, respectively.

```
R> if (exists("clRes")) {
   clusters <- cutree(clRes@clusterObjs$hierarchical, 6)
   ## Scales the average expression values
   aveExpr <- t(scale(t(aveExpr)))
   colnames(aveExpr) <- c("GD-12", "GD-13", "GD-14")
   clustPlot(clusters, aveExpr, 3, 2)
 }
```

7 Determining miRNA Target Genes

To follow-up the results from the differentially expression and clustering analysis, the next step is to determine putative regulatory targets of the differentially expressed miRNAs. To illustrate, we identify the putative targets of the miRNAs contained in the first cluster in Figure **??**. The miRNAs in the first cluster are evaluated for putative targets using the databases TargetScan [3] (package *targetscan.Mm.eg.db*) and miRBase [4] (package *microRNA*). The mouse specific miRNA names are first extracted and then converted to the standard nomenclature using the function `miRNames`, which is included in the accompanying `R` script.

```
R> ## Pull out the MMU specific names
R> if (exists("clusters")) {
   ids1 <- names(clusters[which(clusters==1)])
 } else {
   ids1 <- c("35816", "35861", "35886", "35817", "39256", "38722", "39370",
             "38559", "40185", "35849", "35884", "40069", "39153", "39157",
             "39361", "39299", "35863", "39294", "38316", "39211", "40190",
             "38319", "38995", "35855", "38796", "35899", "39212", "38508",
             "39178", "35889", "38849", "39209")
 }
R> miRs1 <- miRNames(ids1, avedata$genes$Name, avedata$genes$'Gene ID')
R>
```

Targetscan targets are obtained using the code below. The objects in the *targetscan.Mm.eg.db* package are `Bimap` objects, which are mappings from one set of keys (the left keys or `Lkeys`)

[3] TargetScanHuman 5.1, `http://www.targetscan.org/`

[4] MicroCosm Targets, `http://www.ebi.ac.uk/enright-srv/microcosm/htdocs/targets/v5/`

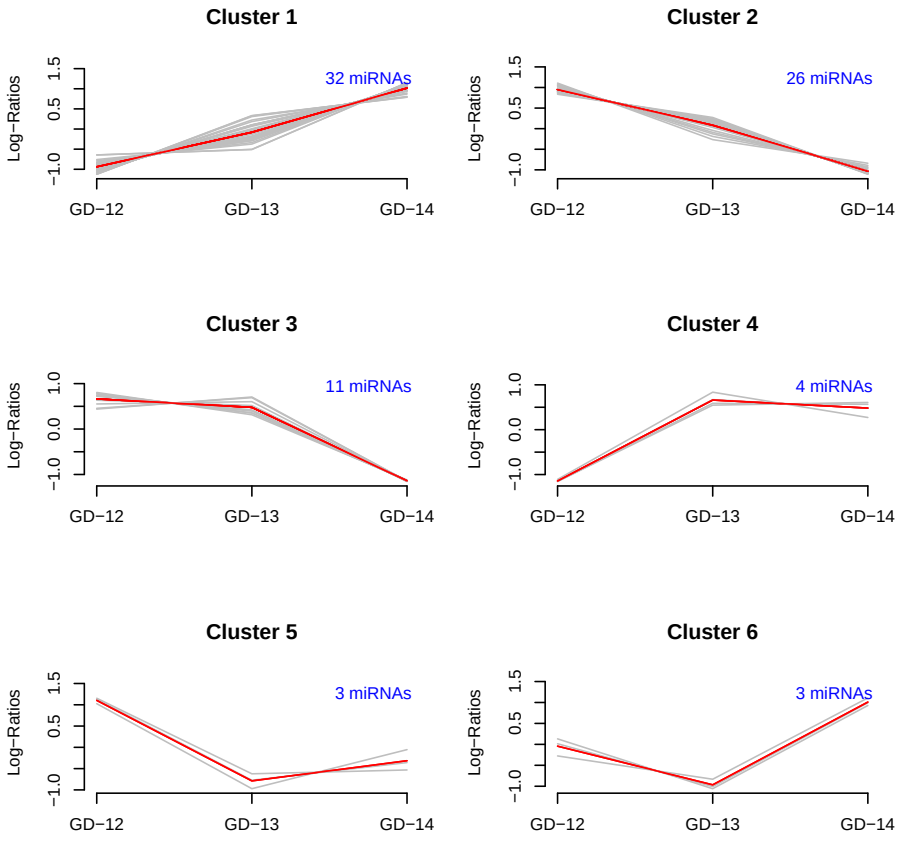

Figure 9: Plot of clustering results for all differentially expressed miRNAs, based on hierarchical clustering with six clusters.

to another (the right keys or `Rkeys`). We start by mapping the miRBase identifiers to their miRNA family names, then map the miRNA families to Entrez Gene identifiers of the targets in the TargetScan database. Several of the miRNAs of interest required slight modifications to their names prior to their mapping.

```
R> if (require("targetscan.Mm.eg.db")) {

   ## These are 'miRNAAnnDbBimap' - Bi-mappings
   res01 <- miRs1%in%ls(targetscan.Mm.egMIRNA) ## only two false
   miRs1[!res01] ## "mmu-miR-126" "mmu-let-7b*"
   miRs1[!res01] <- c("mmu-miR-126-3p", "mmu-let-7b")
   miRs1 <- unique(miRs1) ## have "mmu-let-7b" twice

   miRs1.list <- mget(miRs1, targetscan.Mm.egMIRNA)
   miRs1.fams <- mget(miRs1, targetscan.Mm.egMIRBASE2FAMILY)
   miRs1.targets <- mget(as.character(miRs1.fams), revmap(targetscan.Mm.egTARGETS))
   targets.tscan <- unique(unlist(miRs1.targets))
   length(targets.tscan)
 }

[1] 4640

R>
```

The TargetScan database identifies 4640 unique Entrez Gene identifiers as putative targets.

Mouse miRNA targets in the miRBase database are in the data frame `mmTargets` within the *microRNA* package, and can be obtained using the code below. The targets are stored as Ensembl gene identifiers.

```
R> ######################################################################
R> ## microRNA package
R> ######################################################################
R>
R> if (require("microRNA")) {
   ## Try identifying targets for miRNAs in cluster one ...
   data(mmTargets)
   targets.miRB <- mmTargets$target[which(mmTargets$name%in%miRs1)]
   targets.miRB <- unique(targets.miRB)
   length(targets.miRB)
   ## target ensembl IDs

   ## head(targets.miRB)  ## Ensembl IDs
   ## head(targets.tscan) ## Entrez gene idenifiers
   ## Convert to common naming for use with GSEA, use Entrez Gene IDs
 }
```

```
[1] 13126

R>
```

A total of 13126 Ensembl transcripts are identified as putative targets by miRBase.

Lastly, we take the intersection of the targets from TargetScan and miRBase as our set of putative targets. Ensembl gene identifiers are firstly converted to Entrez Gene identifiers using the *org.Mm.eg.db* Bioconductor package.

```
R> if (require("org.Mm.eg.db")) {
   ## org.Mm.egENSEMBLTRANS   Map Ensembl transcript acession numbers with
   ##                         Entrez Gene identifiers
   idx.miRB <- as.character(targets.miRB)%in%ls(revmap(org.Mm.egENSEMBLTRANS))
   targets.miRB.list <- as.character(targets.miRB)[idx.miRB]
   targets.miRB.entrez <- unlist(mget(targets.miRB.list, revmap(org.Mm.egENSEMBLTRANS)))
   targets.intsect <- intersect(targets.tscan, targets.miRB.entrez)
   length(targets.intsect)
 }

[1] 660

R>
```

The final list contains 660 Entrez Gene identifiers.

8 Gene Set Analysis

As a final step in our analysis, we take the putative miRNA targets from the intersection of the TargetScan and miRBase databases and perform gene set enrichment analysis on them, using the hypergeometric test from the *GOstats* package [**?**]. Terms in the GO hierarchy are analyzed for over-reprensetation of genes from our miRNA target list, relative to the total number from the mouse genome having that annotation. A `GOHyperGParams` object is created which contains the list of targets (`selectedEntrezIds`), the gene "universe" (`entrezUniverse`), the annotation database to use, the GO ontology, and direction and significance level of the test.

```
R> ## Steps:
R> ## 1. Define gene universe (a vector of Entrez Gene IDs).
R> ## 2. Select a list of interesting genes (a vector of Entrez Gene ID).
R> ## 3. Create a GOHyperGParams object.
R> ## 4. Outputs and summary.
R>
R> if (require("GOstats")) {
   selectedEntrezIds <- targets.intsect
   ## Universe =  all entrez gene ids
```

```
   entrezUniverse <- unlist(ls(org.Mm.egENSEMBLTRANS))

   hgCutoff <- 0.001
   GOparams <- new("GOHyperGParams",
                   geneIds=selectedEntrezIds,
                   universeGeneIds=entrezUniverse,
                   annotation="org.Mm.eg", ## may need to install org.Mm.eg.db ...
                   ontology="BP",
                   pvalueCutoff=0.001,
                   conditional=TRUE,
                   testDirection="over")
 }
R>
```

After the `GOHyperGParams` object has been created, the test can be conducted using the `hyperGTest` function. An html file summarizing the results can be created using the `htmlReport` function (regeneration of the report is suppressed here due to time considerations). Particular categories of interest include GO:0060021 (palate development), GO:0048008 (platelet-derived growth factor receptor signaling pathway), GO:0060429 (epithelium development), GO:0030855 (epithelial cell differentiation), GO:0016331 (morphogenesis of embryonic epithelium), GO:0016055 (Wnt receptor signaling pathway), GO:0060828 (regulation of canonical Wnt receptor signaling pathway), GO:0008277 (regulation of G-protein coupled receptor protein signaling pathway), and GO:0007179 (transforming growth factor beta receptor signaling pathway).

```
R> if (exists("GOparams")) {
   hgOver <- hyperGTest(GOparams)
   class(hgOver)
   summary(hgOver)
   ## HTML report of results
   htmlReport(hgOver, file="hgResult.html")
 }
```

As a final step, we evaluate the mature miRNA sequences and seed regions of the miRNAs which target the genes in a particular GO category. To illustrate, the GO category 0007179, transforming growth factor beta receptor signaling pathway, is used. Entrez Gene IDs belonging to this category are identified, and intersected with the selected Entrez Gene IDs corresponding to cluster one of Figure **??**.

```
R> ## Look at GO:0007179 (transforming growth factor beta receptor signaling pathway)
R> ## 1. Find genes
R> if (require("org.Mm.eg.db")) {
   egIdsAll <- get("GO:0007179", org.Mm.egGO2ALLEGS)
   egIds <- intersect(egIdsAll, selectedEntrezIds)
   length(egIds)
```

```
   ## eliminates redundancies (genes w/multiple evidence codes)
 }

[1] 7

R>
```

Next, these 7 Entrez Gene IDs are reverse mapped back to the set of miRNAs which putatively target these genes.

```
R> if (require("targetscan.Mm.eg.db")) {
   ## 2. Find miRNAs which target these genes
   miRs.BetaR.TS <- mget(egIds, targetscan.Mm.egTARGETS)
   miRs.BetaR.fams <- intersect(miRs1.fams, unlist(miRs.BetaR.TS))
   miRs.BetaR.list <- mget(miRs.BetaR.fams, revmap(targetscan.Mm.egMIRBASE2FAMILY))
   miRs.BetaR.mmu <- grep("mmu", unlist(miRs.BetaR.list), value=TRUE)
   ## Now restrict these to just miRs of interest
   miRs.BetaR.clust1 <- intersect(miRs1, miRs.BetaR.mmu)
   length(miRs.BetaR.clust1)

 }

[1] 12

R>
```

Lastly, the mature sequences and seed regions of these 12 miRNAs are determined, using the `mmSeqs` database and `seedRegions` function in package *microRNA*. These sequences can be evaluated for any commonalities, to be used in determining potential targets for follow-up luciferase assays and other functional experiments [**?**]. In this case, the sequences are rather heterogeneous, although the seed region "GAGGUA" does show up in four of the 12 identified miRNAs.

```
R> ## 3. Now look at sequences and seed regions ...
R> if (require("microRNA")) {
   data(mmSeqs)
   idx.betaR <- which(names(mmSeqs)%in%miRs.BetaR.clust1)
   mmSeqs[idx.betaR]
   table(seedRegions(mmSeqs[idx.betaR]))
 }

AACACU AGCUGC CCCUGA GAGGUA GUAAAC UCAAGU UCCAGU UUGGUC
     1      1      1      4      1      1      1      2
```

9 Session Info

```
R> sessionInfo()

R version 3.3.0 RC (2016-04-26 r70550)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.9.5 (Mavericks)

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats4    parallel  stats     graphics  grDevices utils
[7] datasets  methods   base

other attached packages:
 [1] GOstats_2.38.0            graph_1.50.0
 [3] Category_2.38.0           Matrix_1.2-6
 [5] org.Mm.eg.db_3.3.0        microRNA_1.30.0
 [7] targetscan.Mm.eg.db_0.6.1 AnnotationDbi_1.34.0
 [9] IRanges_2.6.0             S4Vectors_0.10.0
[11] clValid_0.6-6             cluster_2.0.4
[13] multtest_2.28.0           latticeExtra_0.6-28
[15] RColorBrewer_1.1-2        MmPalateMiRNA_1.22.0
[17] vsn_3.40.0                lattice_0.20-33
[19] statmod_1.4.24            limma_3.28.0
[21] xtable_1.8-2              Biobase_2.32.0
[23] BiocGenerics_0.18.0

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.4.5          BiocInstaller_1.22.0
 [3] plyr_1.8.3             XVector_0.12.0
 [5] class_7.3-14           tools_3.3.0
 [7] zlibbioc_1.18.0        annotate_1.50.0
 [9] RSQLite_1.0.0          preprocessCore_1.34.0
[11] gtable_0.2.0           DBI_0.4
[13] genefilter_1.54.0      Biostrings_2.40.0
[15] grid_3.3.0             GSEABase_1.34.0
[17] RBGL_1.48.0            XML_3.98-1.4
[19] survival_2.39-2        GO.db_3.3.0
[21] ggplot2_2.1.0          scales_0.4.0
[23] splines_3.3.0          MASS_7.3-45
[25] AnnotationForge_1.14.0 colorspace_1.2-6
[27] affy_1.50.0            munsell_0.4.3
[29] affyio_1.42.0
```

10 List of abbreviations

DIANA: Divisive Analysis

GD: Gestational Day

GEO: Gene Expression Omnibus

GO: Gene Ontology

KEGG: Kyoto Encyclopedia of Genes and Genomes

miRNA: microRNA

PAM: Partitioning Around Medoids

SAM: Significant Analysis of Microarrays

SOM: Self-Organizing Maps

SOTA: Self-Organizing Tree Algorithm

UR: Universal Reference

VSN: Variance Stabilizing Normalization