Assessing signal/noise ratio before and after normalization

Wolfgang Huber

March 13, 2006

Contents

1 Introduction

The purpose of this document is to assess the performance of the probe-response normalization by the function `normalizeByReference` in the *tilingArray* package. We use the example data from the David et al. [1] paper, which is provided in the *davidTiling* package.

```
> library("tilingArray")
> library("davidTiling")
> data("davidTiling")
```

It contains 8 arrays with 6553600 features each. Three of them were hybridized to genomic DNA, which will use a reference for the normalization, and five to RNA.

```
> dim(exprs(davidTiling))
```

```
[1] 6553600       8
```

```
> sampleNames(davidTiling)
```

```
[1] "09_11_04_S96_genDNA_16hrs_45C_noDMSO.cel"
[2] "041119_S96genDNA_re-hybe.cel"
[3] "041120_S96genDNA_re-hybe.cel"
[4] "05_04_27_2xpolyA_NAP3.cel"
[5] "05_04_26_2xpolyA_NAP2.cel"
[6] "05_04_20_2xpolyA_NAP_2to1.cel"
[7] "050409_totcDNA_14ug_no52.cel"
[8] "030505_totcDNA_15ug_affy.cel"
```

2 PM and MM features

First, we determine the indices of the PM and MM features. The array has 2560 rows and 2560 columns. If we count the rows and columns from 0 to 2559, then the indices of the features' intensities in the expression matrix `exprs(davidTiling)` are given by r*2560+c. The PM features lie in rows $1, 3, \ldots, 2557$, their corresponding MM features in rows $2, 4, \ldots, 2558$:

```
> nc = as.integer(2560)
> PMind = rep(seq(as.integer(1), nc - as.integer(3), by = as.integer(2)),
+     each = nc) * nc + (1:nc)
> MMind = PMind + nc
```

To verify this, let's look at the scatterplot of PM versus MM values for the first chip, shown in Figure 1:

```
> x1 = log(exprs(davidTiling)[PMind, 1], 2)
> x2 = log(exprs(davidTiling)[MMind, 1], 2)
> smoothScatter(x1, x2, nrpoints = 0, xlab = "PM", ylab = "MM")
> abline(a = 0, b = 1, col = "red")
```

3 Background features

For the background estimation, we need to specify a set of "background" features, that is features for which we expect no specific signal. The feature information in the *davidTiling* package is stored in the environment `probeAnno`. The following code selects all probe that do not map to a genomic feature (such as ORF, ncRNA) on either strand.

```
> data("probeAnno")
> ispm = rep(FALSE, nc * nc)
> ispm[PMind] = TRUE
> isbg = (probeAnno$probeReverse$no_feature == "no" & probeAnno$probeDirect$no_feature ==
+     "no" & ispm)
```

4 Normalization

```
> isRNA = davidTiling$nucleicAcid %in% c("poly(A) RNA", "total RNA")
> isDNA = davidTiling$nucleicAcid %in% "genomic DNA"
> stopifnot(sum(isRNA) == 5, sum(isDNA) == 3)
> xn2 = cache("xn2", normalizeByReference(davidTiling[, isRNA],
+     davidTiling[, isDNA], pm = PMind, background = isbg, plotFileNames = sprintf("assessNorm-n
+         seq(along = which(isRNA)))))
```

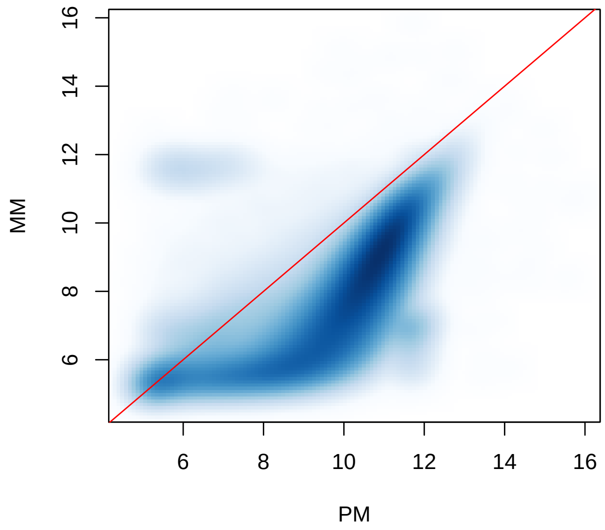

Figure 1: Scatterplot of M versus MM values for the first chip in the `davidTiling` data.

5 Alternative normalization methods

5.1 Without dropping the worst 5% probes

For comparison, we also compare to the situation in which we do not throw out the weakest features, by setting `cutoffQuantile=0`.

```
> xn1 = cache("xn1", normalizeByReference(davidTiling[, isRNA],
+     davidTiling[, isDNA], pm = PMind, background = isbg, cutoffQuantile = 0))
```

5.2 PM-MM

```
> z = exprs(davidTiling)[PMind, isRNA] - exprs(davidTiling)[MMind,
+     isRNA] + 16
> z[z <= 0] = NA
> xMM = matrix(as.numeric(NA), nrow = nc * nc, ncol = sum(isRNA))
> xMM[PMind, ] = log2(z)
```

6 Assessment

6.1 Visually

We would like to visualize the data along genomic coordinates. We select the features that map to the longest chromosome, which is chromosome 4. The integer vectors `sta` and `end` contain the start and end coordinate of their match, `ind` their indices in the array `exprs(davidTiling)`. hit.

```
> sta = probeAnno$"9.-.start"
> end = probeAnno$"9.-.end"
> ind = probeAnno$"9.-.index"
```

We construct a list of vectors, each containing different versions of the intensity data, in order that corresponds to `sta` and `ind` from above.

```
> dat = vector(mode = "list", length = 5)
> dat[[1]] = log2(exprs(davidTiling)[ind, which(isDNA)[1]])
> dat[[2]] = log2(exprs(davidTiling)[ind, which(isRNA)[1]])
> dat[[3]] = xMM[ind, 1]
> dat[[4]] = dat[[2]] - dat[[1]]
> dat[[5]] = exprs(xn1)[ind, 1]
> dat[[6]] = exprs(xn2)[ind, 1]
> for (j in 4:length(dat)) dat[[j]] = dat[[j]] - quantile(dat[[j]],
+     0.05, na.rm = TRUE)
```

We select a 10kB region around the highly expressed genes RPN2 and SER33 to fit on a plot, and set the y-axis limits:

```
> sel = (sta >= 216600 & end <= 227000)
> ysc = sapply(dat, function(py) quantile(py, probs = c(0, 0.99),
+     na.rm = TRUE))
> ysc[, 4] = ysc[, 5] = ysc[, 6]
```

Now we are ready to plot:

```
> anno = data.frame(start = c(217860, 221078), end = c(220697,
+     222487), name = I(c("RPN2", "SER33")))
> ticks = c(217, 223, 224, 225, 226)
> comparisonPlot((sta + end)[sel]/2, lapply(dat, "[", sel), yscale = ysc,
+     anno = anno, ticks = ticks)
```

6.2 Quantitatively

```
> positiveCtrls = cbind(c(217860, 220697), c(221078, 222487))
> negativeCtrls = cbind(c(216800, 217700), c(222800, 227000))
```

This function calculates number that quantify *signal* and *noise*. Noise is calculated as the weighted average of the differences between 95% and 5% quantiles of the data within each of the control regions. Divide by $Q^{\mathrm{Norm}}_{0.95} - Q^{\mathrm{Norm}}_{0.05} \approx 3.28$ so that it corresponds to the standard deviation if the distribution were Normal.

```
> fac = 2 * qnorm(0.95)
> withinAndBetween = function(x) {
+     meanAndSd = function(region) {
+         d = x[(sta >= region[1]) & (end <= region[2])]
+         c(mean(d, na.rm = TRUE), diff(quantile(d, c(0.05, 0.95),
+             na.rm = TRUE))/fac, sum(!is.na(d)))
+     }
```

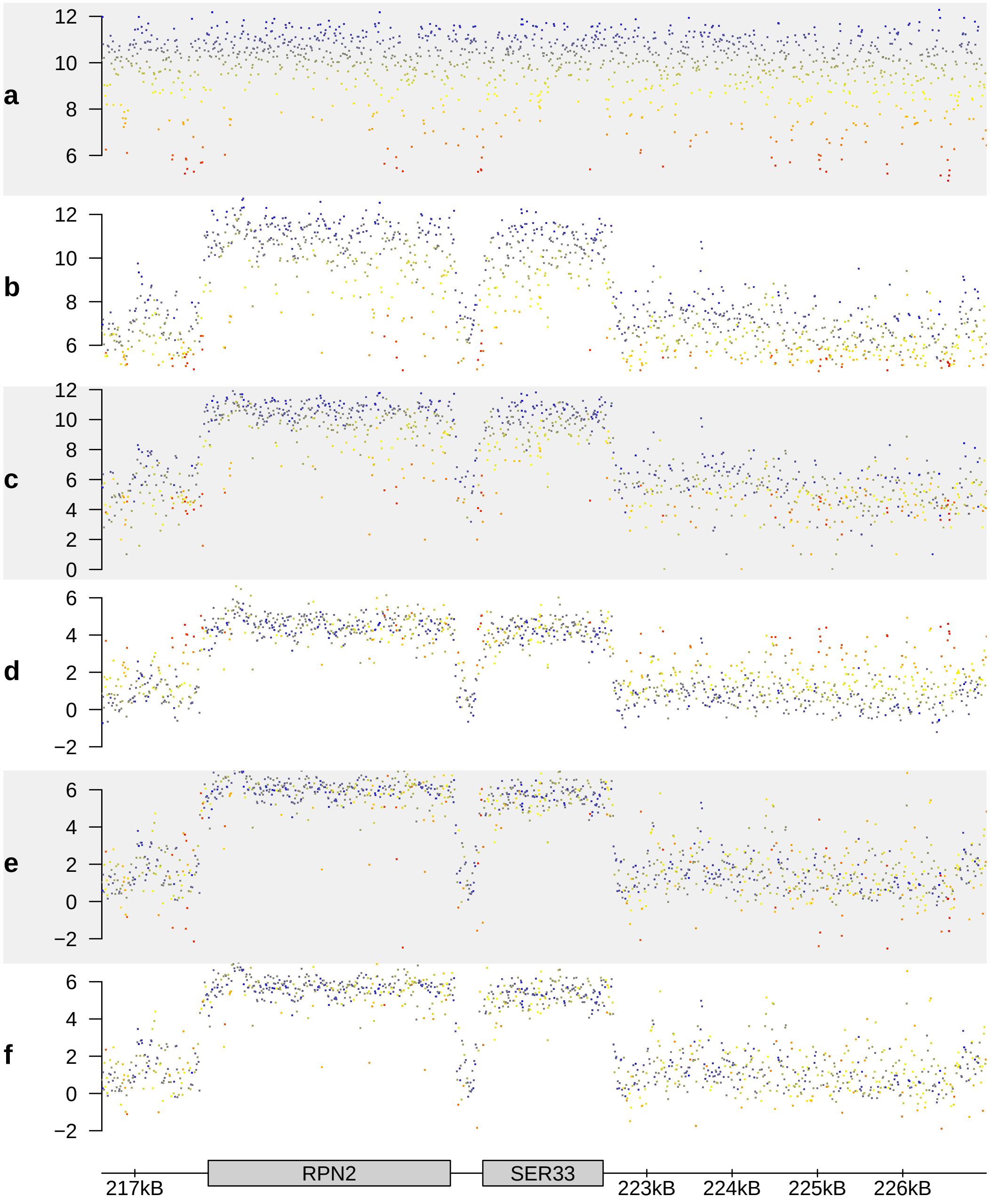

Figure 2: Scatterplot of different types of signal (y-axis) along genomic coordinates (x-axis). Each dot corresponds to a microarray feature. a) signal from one of the DNA hybridizations (logarithmic scale, base 2). The y-coordinate of each dot is also encoded using a pseudo-color scheme. Dark red corresponds to features that have a very weak response, dark blue to those with the strongest response. The same coloring is also used in panels b)-f). b) unnormalized intensities from one of the poly(A) RNA bybridizations (logarithmic scale, base 2). c) $\log_2(\mathrm{PM} - \mathrm{MM} + 16)$ d) Divide RNA-signal by DNA-signal then take logarithm (base 2). e) Background subtraction of the RNA-signal, divide by DNA-signal, then variance stabilizing normalization (vsn, glog base 2). f) In addition to d), drop the 5% weakest features in the DNA hybridization.

```
+     p = apply(positiveCtrls, 2, meanAndSd)
+     n = apply(negativeCtrls, 2, meanAndSd)
+     wtav = function(q, i) sum(q[i, ] * q[3, ])/sum(q[3, ])
+     signal = sum(p[1, ] * p[3, ])/sum(p[3, ]) - sum(n[1, ] *
+         n[3, ])/sum(n[3, ])
+     noise = (sum(p[2, ] * p[3, ]) + sum(n[2, ] * n[3, ]))/(sum(p[3,
+         ]) + sum(n[3, ]))
+     return(c("s/n" = signal/noise, s = signal, n = noise))
+ }
```

```
> sn = sapply(dat, withinAndBetween)
```

```
     [,1] [,2] [,3]  [,4]  [,5]  [,6]
s/n 0.419 3.17 3.41 3.520 4.470 4.800
s   0.562 3.67 4.64 3.109 4.318 4.296
n   1.341 1.16 1.36 0.883 0.966 0.895
```

References

[1] Lior David, Wolfgang Huber, Marina Granovskaia, Joern Toedling, Curtis J. Palm, Lee Bofkin, Ted Jones, Ronald W. Davis, and Lars M. Steinmetz A high-resolution map of transcription in the yeast genome. *PNAS*, 2006. 1