Robust calibration and variance stabilization with VSN

Wolfgang Huber

October 27, 2003

Introduction

`vsn` is a method to preprocess DNA microarray intensity data. As input, the function `vsn` takes the raw intensity measurements from the DNA probes on a series of microarrays. The intensities from each array are calibrated by a suitable affine transformation, then transformed by a variance-stabilizing transformation. After this, systematic array- or dye-biases should be removed, and the variance should be approximately independent of the mean intensity. This is useful for subsequent analyses such as hypothesis tests, ANOVA modeling, clustering, or classification that assume that the variance is the same for all observations[1]. Differences between the transformed values are the so-called "generalized log-ratios". If both numerator and denominator are well above background, generalized log-ratios coincide with the usual log-ratios: $h(x_i) - h(x_j) \approx \log(x_i) - \log(x_j) = \log(x_i/x_j)$ if $x_i, x_j \gg 0$ [1, 3]. In contrast to log-ratios, they remain well-defined and statistically meaningful if x_i or x_j are close to zero.

1 The data

The prefered input type for `vsn` are objects of class `exprSet`. It is also possible to pass matrices, data frames with numeric columns only, and objects of class `marrayRaw`. Furthermore, `vsn` can be used as a normalization method in the function `expresso` in the package `affy` (cf. Section 6).

To load intensity data from your own experiments, you can use the function `read.table`, the `read`-functions from the package `marrayInput`, or `ReadAffy` from the package `affy`.

2 Running vsn on the data from a single cDNA array

The package includes example data from a cDNA array on which two biologically highly similar samples, one labeled in green (Cy3), one in red (Cy5), were hybridized.

```
> library(vsn)
> data(kidney)
```

The two columns of the matrix `exprs(kidney)` contain the green and red intensities, respectively. Let's try out `vsn` on these example data. In Fig. 1 you can see the scatterplot of the calibrated and transformed data. For comparison, the scatterplot of the log-transformed raw intensities is also shown.

[1]Note that `vsn` only addresses the dependence of the variance on the mean intensity. There may be other factors influencing the variance, such as gene-inherent properties, or changes of the tightness of transcriptional control in different conditions. If necessary, these need to be addressed by other methods.

```
> par(mfrow = c(1, 2))
> nkid <- vsn(kidney)

vsn: 8704 x 2 matrix (lts.quantile=0.5). Please wait for 11 dots:
...........

> plot(exprs(nkid), main = "vsn", pch = ".")
> plot(log.na(exprs(kidney)), main = "raw", pch = ".")
```

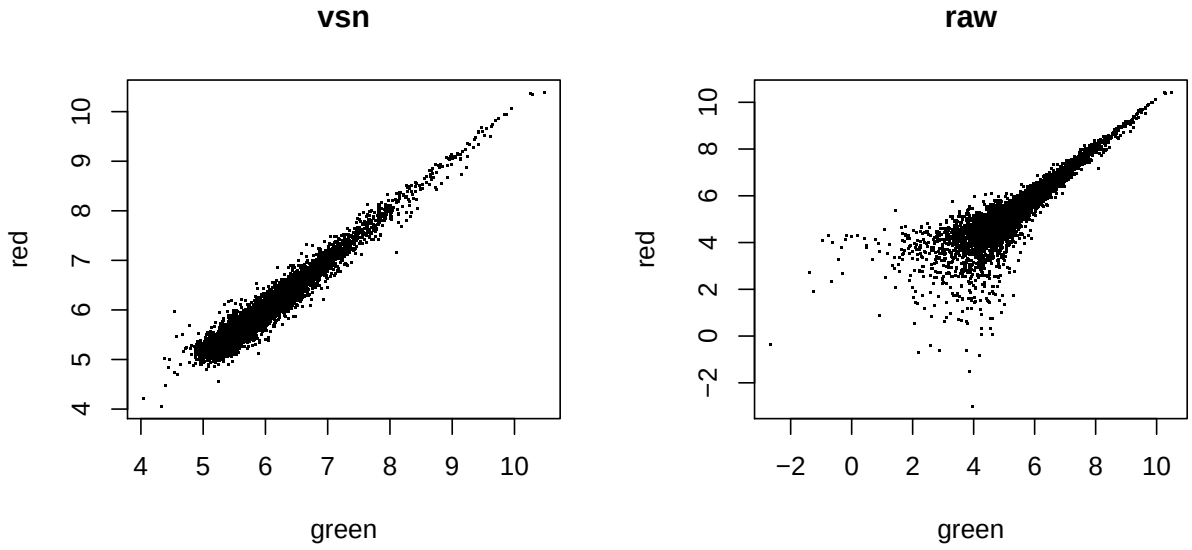

Figure 1: Scatterplots of the kidney example data

`vsn` returns the transformed intensities in an object of class `exprSet`. Its slot `exprs` is a matrix of the same size as the input data. The plot in Fig. 1 shows the complete set of $n = 9216$ red and green intensities, without any thresholding or masking of data points. To verify the variance stabilization, there is the function `meanSdPlot`. For each probe $k = 1, \ldots, n$ it shows the estimated standard deviation $\hat{\sigma}_k$ on the y-axis versus the rank of the average $\hat{\mu}_k$ on the x-axis,

$$\hat{\mu}_k = \frac{1}{d}\sum_{i=1}^{d} h_{ki} \qquad \hat{\sigma}_k^2 = \frac{1}{d-1}\sum_{i=1}^{d}(h_{ki} - \hat{\mu}_k)^2. \tag{1}$$

```
> par(mfrow = c(1, 2))
> meanSdPlot(nkid, ranks = TRUE)
> meanSdPlot(nkid, ranks = FALSE)
```

Such a plot is shown in Fig. 2. The red dots, connected by lines, show the running median of the standard deviation[2]. Within each window, the median may be considered a pooled estimator of the standard deviation, and the curve given by the red line is an estimate of the systematic dependence of the standard deviation on the mean. After variance stabilization, this should be approximately

[2]Window width: 10%, window midpoints 5%, 10%, 15%, ...

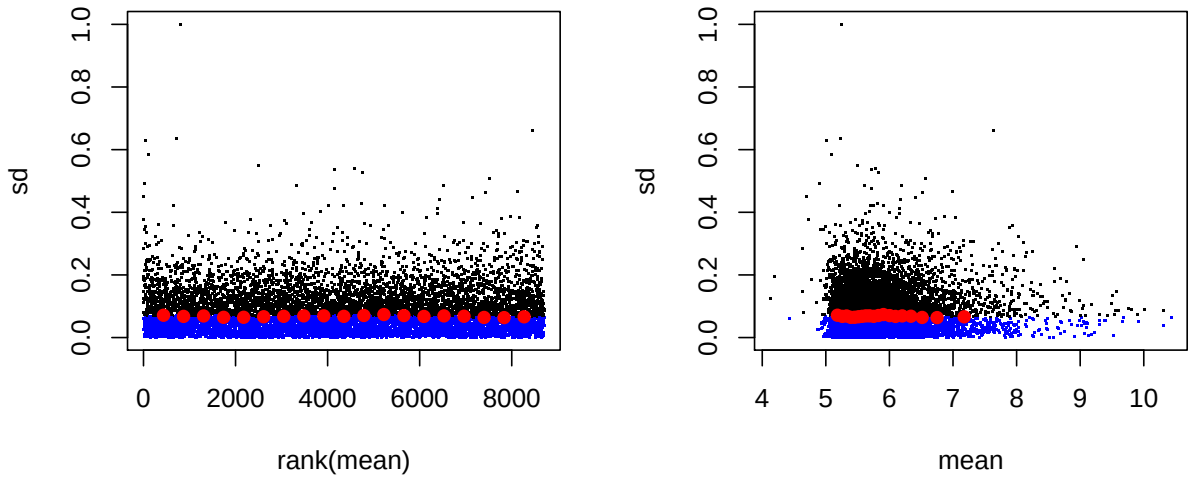

Figure 2: Standard deviation versus rank of the mean, and the mean, respectively

a horizontal line. It may have some random fluctuations, but should not show an overall trend. If this is not the case, that could indicate a data quality problem, see Section 8. The rank ordering distributes the data evenly along the x-axis, which is in many cases useful for the visualization. A plot in which the x-axis shows the average intensities themselves is obtained by calling the `plot` command with the argument `ranks=FALSE`.

The parameter estimation in `vsn` works in an iterative manner. To verify that the iterations have converged, you can call the function `vsnPlotPar`.

```
> par(mfrow = c(1, 2))
> vsnPlotPar(nkid, "offsets")
> vsnPlotPar(nkid, "factors")
```

The plots in Fig. 3 show the values of the estimated calibration and variance stabilization parameters on the y-axis as a function of the iteration index. All curves should reach a plateau well before the last iteration. If this is not the case, the number of iterations may be increased through the parameter `iter`, and/or `lts.quantile` may be decreased. It could also indicate a data quality problem, see Section 8.

The "generalized log-ratios" [1] for this experiment may be obtained for further processing through

```
> M <- exprs(nkid)[, 2] - exprs(nkid)[, 1]
> hist(M, breaks = 50, col = "#d95f0e")
```

The histogram is shown in Fig. 3.

3 Calibration

We can access the transformation and calibration parameters through

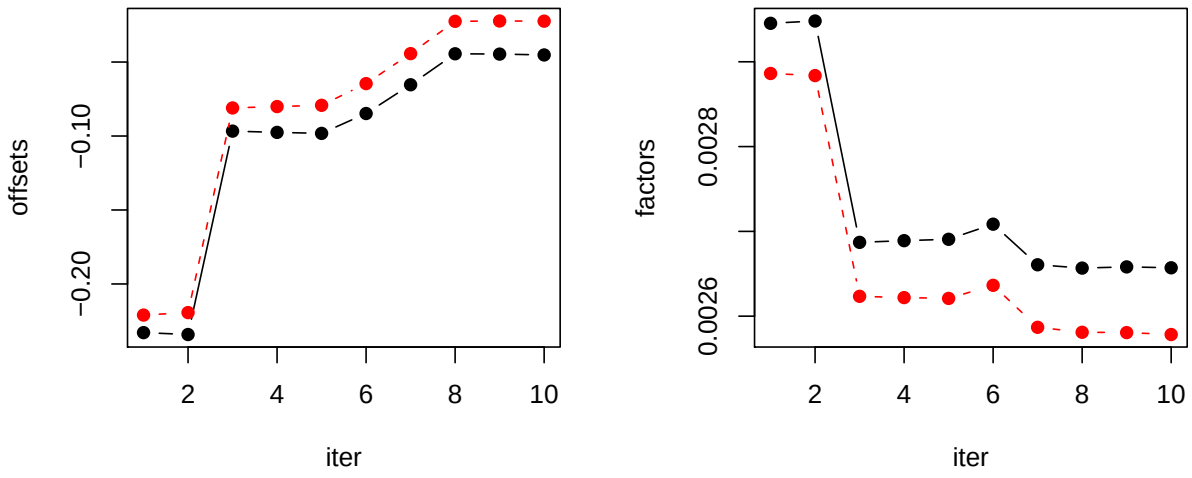

Figure 3: Iteration trajectory of the calibration and transformation parameters

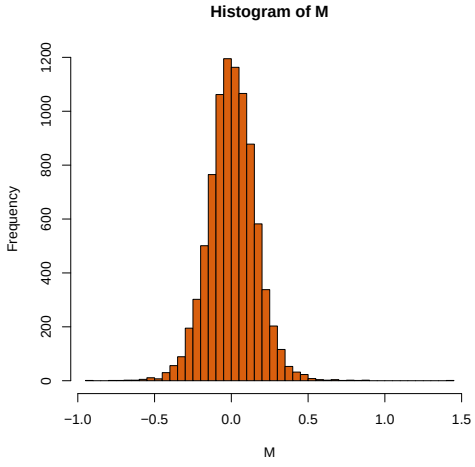

Figure 4: Histogram of generalized log-ratios for the kidney example data

```
> prep <- preproc(description(nkid))
> names(prep)

[1] "vsnParams"        "vsnParamsIter"    "vsnTrimSelection"

> prep$vsnParams

       offs1        offs2         fac1         fac2
-0.045192754 -0.022364154  0.002657068  0.002578388
```

The `description` slot of an `exprSet` is an object of class `MIAME`, and may contain annotation (or "metadata") pertinent to the experiment represented by the object. For an exprSet with d columns, `prep$vsnParams` is a numeric vector of length $2d$. Its elements $1, \ldots, d$ contain the additive calibration and transformation parameters a_i, its elements $d+1, \ldots, 2d$ the multiplicative ones. Compare the above numbers with the final values in Fig. 3.

If y_{ki} is the matrix of uncalibrated data, with k indexing the rows and i the columns, then the calibrated data y'_{ki} is obtained through scaling by b_i and shifting by a_i:

$$y'_{ki} = a_i + b_i y_{ki} \tag{2}$$

Now suppose the kidney example data were not that well measured, and the red channel had a baseline that was shifted by 500 and a scale that differed by a factor of 0.25:

```
> bkid <- kidney
> exprs(bkid)[, "red"] <- 0.25 * (500 + exprs(bkid)[, "red"])
```

We can again call `vsn` on this data

```
> nbkid <- vsn(bkid)

> par(mfrow = c(1, 2))
> plot(exprs(bkid), main = "raw", pch = ".", log = "xy")
> plot(exprs(nbkid), main = "vsn", pch = ".")
> preproc(description(nbkid))$vsnParams

       offs1        offs2         fac1         fac2
-0.178307013 -1.542177835  0.002856737  0.011104310
```

The factor for the red channel is now about four times as large as before. The result is shown in Fig. 5.

4 Running vsn on the data from multiple cDNA arrays

The package includes example data from a series of 8 cDNA arrays on which different lymphoma were hybridized together with a reference cDNA [7].

```
> data(lymphoma)
> pData(lymphoma)
```

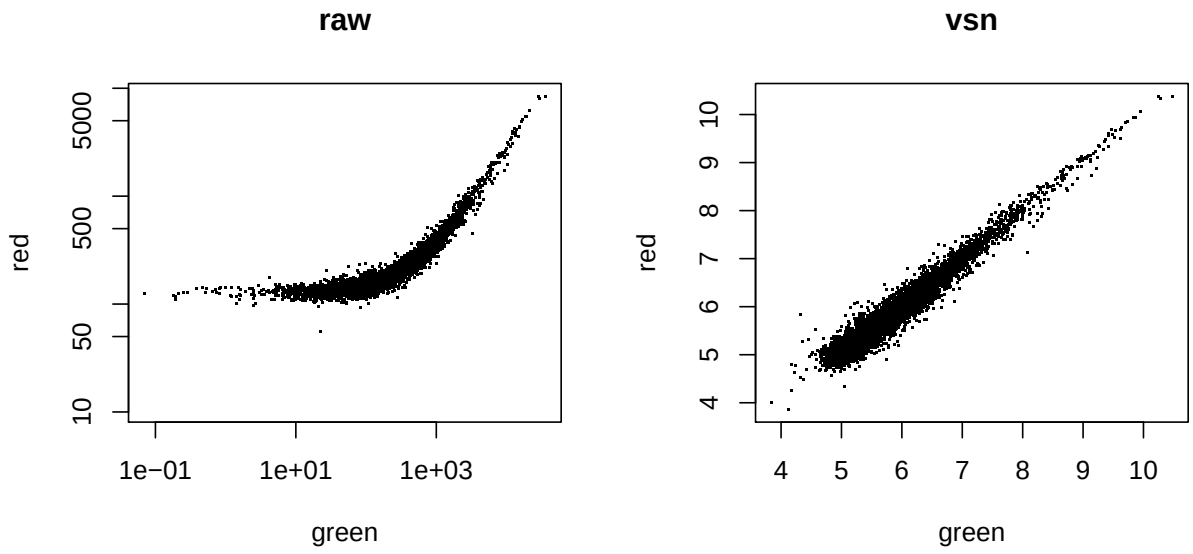

Figure 5: Scatterplots for badly biased data. Left hand side: raw data on log-log scale, right hand side: after calibration and transformation with vsn.

```
      name    sample
1  lc7b047 reference
2  lc7b047    CLL-13
3  lc7b048 reference
4  lc7b048    CLL-13
5  lc7b069 reference
6  lc7b069    CLL-52
7  lc7b070 reference
8  lc7b070    CLL-39
9  lc7b019 reference
10 lc7b019 DLCL-0032
11 lc7b056 reference
12 lc7b056 DLCL-0024
13 lc7b057 reference
14 lc7b057 DLCL-0029
15 lc7b058 reference
16 lc7b058 DLCL-0023
```

The 16 columns of the `lymphoma` object contain the red and green intensities, respectively, from the 8 slides, as shown in the table. Thus, the CH1 intensities are in columns $1, 3, \ldots, 15$, the CH2 intensities in columns $2, 4, \ldots, 16$.

There are now two modes of operation for `vsn`: First, `vsn` can be called for each slide in turn. Second, if there is reason to believe that the slides are of similar quality, it can also be called on all of them at once:

```
> lym <- vsn(lymphoma, verbose = FALSE)
```

```
> meanSdPlot(lym)
```

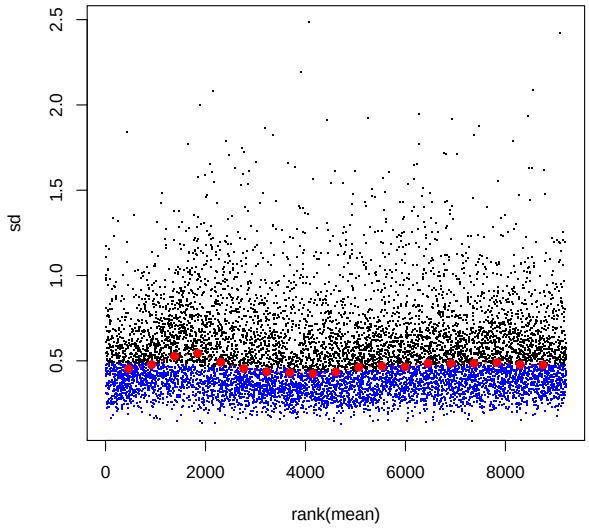

Figure 6: Standard deviation versus rank of the mean for the lymphoma example data

This calculation may take a while. Again, Fig. 6 helps to visually verify that the variance stabilization worked. As above, we can obtain the "generalized log-ratios" for each slide, by subtracting the common reference intensities from those for the 8 samples:

```
> refrs <- (1:8) * 2 - 1
> samps <- (1:8) * 2
> M <- exprs(lym)[, samps] - exprs(lym)[, refrs]
> colnames(M) <- pData(lymphoma)[samps, "sample"]
> A <- rowMeans(exprs(lym))
> par(mfrow = c(1, 2))
> plot(A, M[, "CLL-13"], pch = ".")
> abline(h = 0, col = "red")
> plot(A, M[, "DLCL-0032"], pch = ".")
> abline(h = 0, col = "red")
```

Fig. 7 shows the analagon to the M-vs-A-plots as described in reference [5]. Note that in the left scatterplot, there is a cloud of points at low intensities that is concentrated slightly off the line $M = 0$. In the right scatterplot, a similar cloud sits right on the $M = 0$ line. This could be related to a quality problem with the left slide (e. g. related to the PCR amplification or the printing, see Section 8).

5 Comparing calibration and data transformation methods

To compare different microarray calibration and data transformation methods one needs to specify a measure of goodness. One approach is to compare the obtained values against a known truth. This

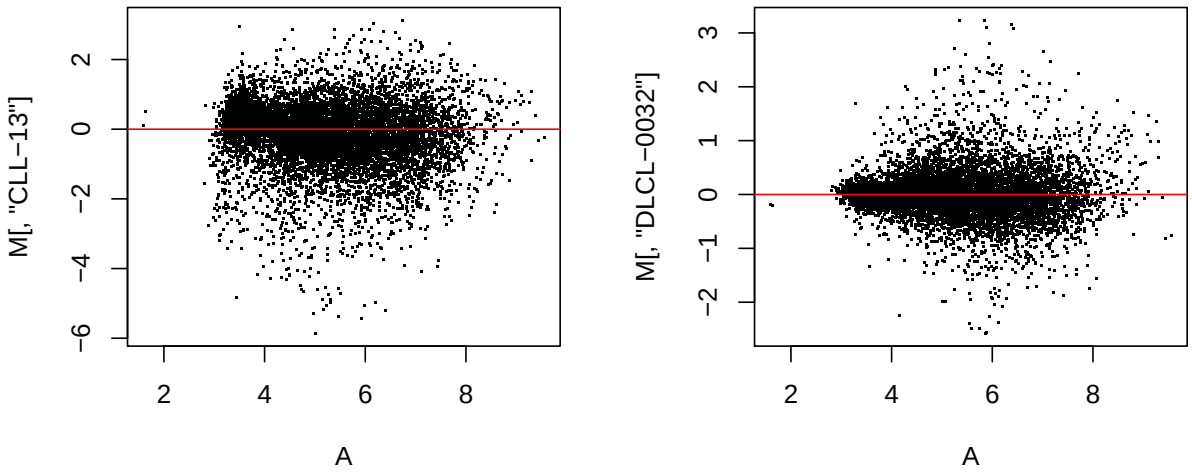

Figure 7: Mean-difference plots for two slides from the lymphoma example data

can be done in controlled spike-in experiments and in dilution series, which allow to systematically assess the performance of the methods at different biologically relevant spike-in concentrations. Like any statistical method, the methods may make different choices with respect to the trade-off between bias and variance.

Here, we just consider a much coarser and simpler criterion: the overall sensitivity and specificity in detecting differential transcription of the combined procedures of data preprocessing and statistical hypothesis testing. Such an analysis can generally be applied to any data set that contains replicated measurements of samples from biologically clearly distinct, known groups (e. g. [1]).

We use the t-test to find genes that are differentially transcribed between CLL and DLCL. The comparison relies on the idea that for a given selection of genes we can estimate the *False Discovery Rate* through a permutation method [6]. Thus, if a method consistently produces a smaller false discovery rate for the same size gene list than another method, it may be considered more specific.

In the following piece of code, data is transformed to log-ratios and normalized by the global median. The result, together with that of the previous section, is stored in the 3-dimensional array `Ms`. Fig. 8 shows, for one of the arrays, a comparison of the resulting log-ratios.

```
> library(marrayNorm)
> Ms <- array(0, dim = c(dim(M), 2))
> Ms[, , 1] <- M
> mr <- new("marrayRaw", maGf = exprs(lymphoma)[, refrs], maRf = exprs(lymphoma)[,
+     samps])
> mn <- maNorm(mr, norm = "median", echo = T)
> Ms[, , 2] <- mn@maM
> plot(Ms[, 1, ], xlab = "vsn", ylab = "global median", main = "M (array 1)",
+     pch = ".")
```

The function `calc.fdr` provides a very simple-minded implementation of the estimation of the

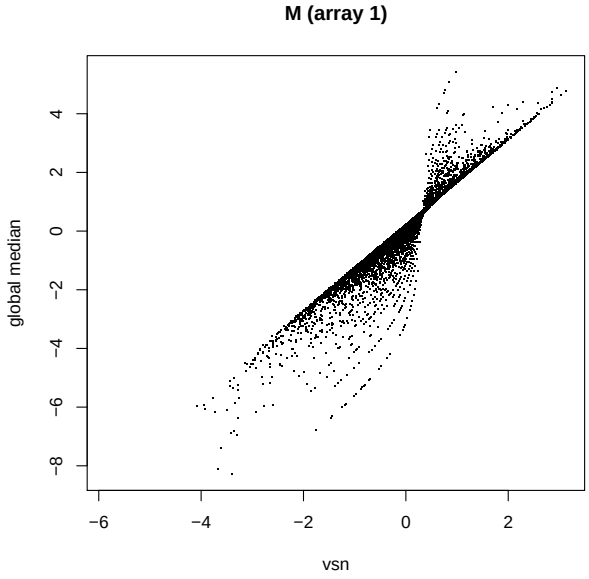

Figure 8: Generalized log-ratio from `vsn` (x-axis) versus the log-ratio after global median normalization (y-axis)

false discovery rate from the permutation distribution of the sorted test statistics. The result is shown in Fig. 9.

```
> library(multtest)
> calc.t <- function(classlabel, M) {
+     t <- mt.teststat(M, classlabel)
+     t[abs(t) > 1e+30] <- NA
+     return(t)
+ }

> calc.fdr <- function(M, classlabel, nrgeneselect) {
+     t <- calc.t(classlabel, M)
+     st <- sort(abs(t), decreasing = TRUE)
+     threshold <- st[nrgeneselect]
+     n <- length(classlabel)
+     grp2 <- which(classlabel)
+     nck <- nchoosek(n - 1, length(grp2) - 1)
+     permclasslabel <- matrix(0, nrow = n, ncol = ncol(nck))
+     for (p in 1:ncol(nck)) {
+         permclasslabel[n, p] <- 1
+         permclasslabel[nck[, p], p] <- 1
+     }
+     permt <- apply(permclasslabel, 2, calc.t, M)
+     fdr <- numeric(length(nrgeneselect))
+     for (j in 1:length(threshold)) {
+         pnrsel <- apply(permt, 2, function(pt) length(which(abs(pt) >=
```

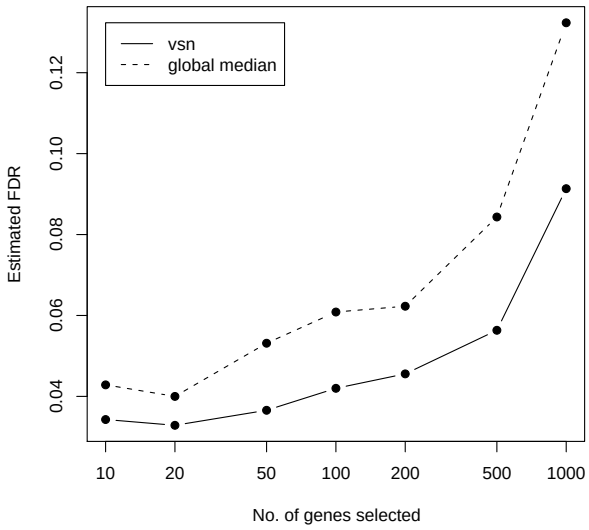

Figure 9: False Discovery Rate

```
+             threshold[j])))
+         fdr[j] <- mean(pnrsel)/nrgeneselect[j]
+     }
+     return(fdr)
+ }

> classlabel <- regexpr("CLL", colnames(M)) > 0
> nrgeneselect <- c(10, 20, 50, 100, 200, 500, 1000)
> fdr <- apply(Ms, 3, calc.fdr, classlabel, nrgeneselect)
> plot(range(nrgeneselect), range(fdr), type = "n", log = "x",
+     xlab = "No. of genes selected", ylab = "Estimated FDR")
> for (j in 1:ncol(fdr)) lines(nrgeneselect, fdr[, j], type = "b",
+     pch = 19, lty = j)
> legend(10, max(fdr), c("vsn", "global median"), lty = 1:ncol(fdr))
```

6 Running vsn on Affymetrix data

The package `affy` provides excellent functionality for reading and processing Affymetrix genechip data. To use `vsn` for the calibration and transformation of the probe intensities, a wrapper for `vsn` is provided that can be used within the data processing routines of `affy`. See the documentation for the package `affy` for more information about data structures and other available methods. There are different ways to treat the mismatch (MM) intensities, to adjust for background, and summarize the probe sets.

The following code is taken from the example in the man page for `normalize.AffyBatch.vsn`. The resulting plot is shown in Fig. 10.

```
> library(affy)
```

```
> library(affydata)
> data(Dilution)
> normalize.AffyBatch.methods <- c(normalize.AffyBatch.methods,
+     "vsn")
> es1 = expresso(Dilution[1:2], bg.correct = FALSE, normalize.method = "vsn",
+     pmcorrect.method = "pmonly", summary.method = "medianpolish")
> es2 = expresso(Dilution[1:2], bgcorrect.method = "rma", normalize.method = "quantiles",
+     pmcorrect.method = "pmonly", summary.method = "medianpolish")
> x1 = exprs(es1)
> x2 = exprs(es2)
> oldpar = par(mfrow = c(2, 2), pch = ".")
> plot(x1, main = "vsn: chip 3 vs 4")
> plot(x2, main = "rma: chip 3 vs 4")
> ylim = c(-0.7, 0.7)
> plot(rank(rowSums(x1)), diff(t(x1)), ylim = ylim, main = "rank(mean) vs differences")
> abline(h = 0, col = "red")
> plot(rank(rowSums(x2)), diff(t(x2)), ylim = ylim, main = "rank(mean) vs differences")
> abline(h = 0, col = "red")
> par(oldpar)
```

7 Verifying and assessing the performance of vsn with simulated data

There are two functions `sagmbSimulateData` and `sagmbAssess` that can be used to generate simulated data and assess the difference between the 'true' and 'estimated' data calibration and transformation by vsn. An example is shown in the code chunk below. Reference [2] describes in more detail (i) the simulation model, (ii) the assessment strategy, and (iii) a comprehensive suite of assessments with respect to the number of probes n, the number of arrays d, the fraction of differentially expressed genes de, and the fraction of up-regulated genes up.

```
> n <- c(500, 1000, 2000, 4000, 8000)
> d <- 2
> de <- c(0, 0.2)
> up <- 0.5
> nrrep <- 8
> res <- array(NA, dim = c(length(n), nrrep, length(de)))
> for (i in seq(along = de)) {
+     for (k in seq(along = n)) {
+         for (r in 1:nrrep) {
+             sim <- sagmbSimulateData(n[k], d, de = de[i], up = 0.5)
+             ny <- vsn(sim$y)
+             res[k, r, i] <- sagmbAssess(exprs(ny), sim)
+         }
+     }
+ }
```

```
> par(mfrow = c(1, 2))
> for (i in seq(along = de)) {
```

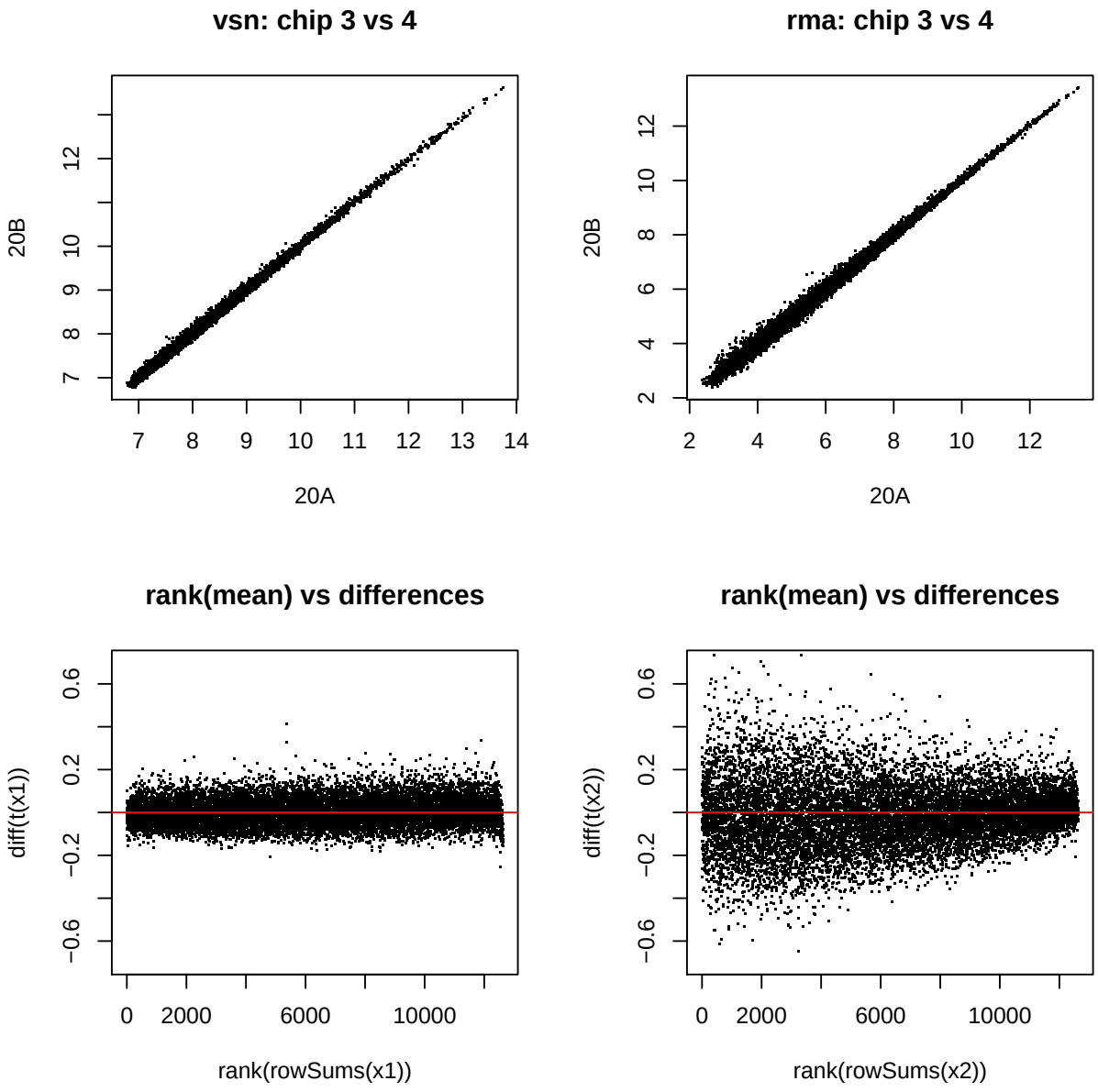

Figure 10: `normalize.AffyBatch.vsn` example

```
+     matplot(n, res[, , i], pch = 20, log = "xy", col = "#909090",
+         main = paste("de=", de[i]))
+     lines(n, rowMeans(res[, , i]), col = "blue")
+ }
```

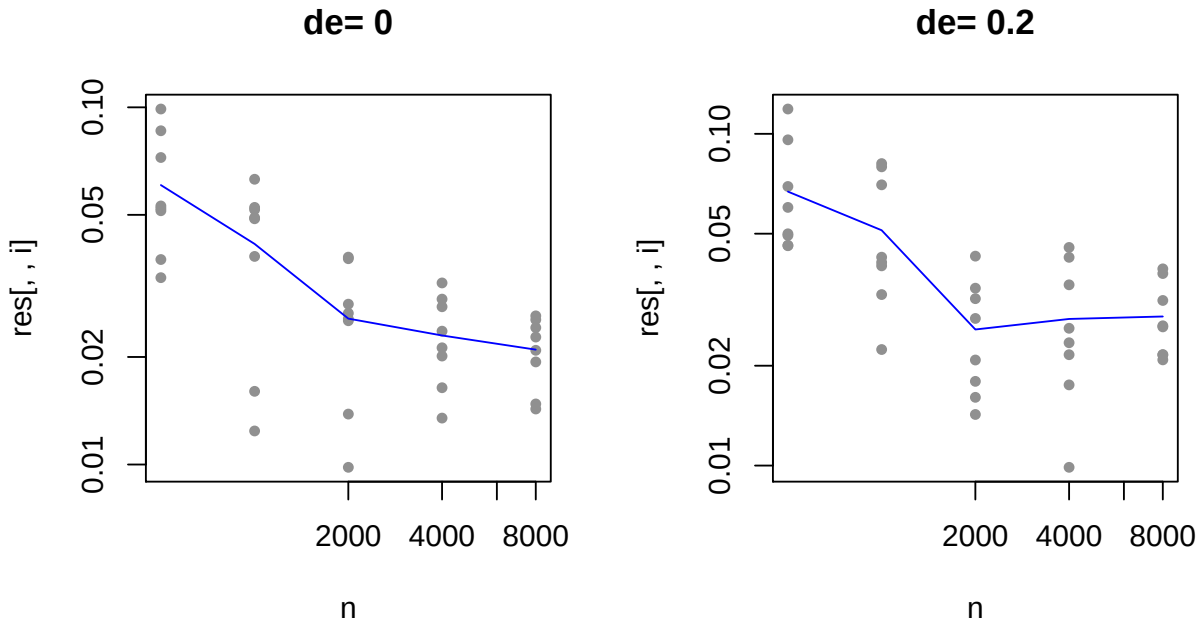

Figure 11: Estimation error for the transformation: the root mean squared difference between true and estimated transformed data, as a function of the number of genes n. If `vsn` works correctly, the estimation error should decrease roughly as $n^{-1/2}$.

8 Quality control

`vsn` makes some assumptions about your data that need to hold if it is to produce meaningful results. We have found them appropriate for many microarray experiments, but it is your responsibility to make sure that they hold for your data.

First, `vsn` assumes that the measured signal y_{ik} increases, to sufficient approximation, proportionally to the mRNA abundance c_{ik} of gene k on the i-th array, or on the i-th color channel:

$$y_{ik} \approx a_i + b_i b_k c_{ik}. \tag{3}$$

For a series of d single-color arrays such as Affymetrix arrays or cDNA nylon membranes, $i = 1, \ldots, d$, and the different factors b_i reflect the different initial amounts of sample mRNA, or different overall reverse transcription, hybridization and detection efficiencies. The probe affinity b_k contains factors that affect all measurements with probe k in the same manner, such as sequence-specific labelling efficiency. The b_k are assumed to be the same across all arrays. There can be a non-zero

overall offset a_i for each color channel. For a two-color cDNA array, $i = 1, 2$, and the b_i take into account the different overall efficiencies of the two dyes[3].

Situations in which the assumption (3) is violated include:

Saturation. The biochemical reactions and/or the photodetection can be run in such a manner that saturation effects occur. It may be possible to rescue such data by using non-linear transformations. Alternatively, it is recommended that the experimental parameters are chosen to avoid saturation.

Print-tip and PCR effects. In cDNA microarray data, systematic patterns have been observed that are associated with the print-tips used for the spotting of DNA, and with the microtitre plates used to store and amplify the DNA. Possibly, these effects could be included in (3) by replacing a_i and b_i by print-tip and/or plate specific coefficients, but presently `vsn` does not include this option.

Batch effects. The probe affinities b_k may differ between different manufacturing batches of arrays due, e.g., to different qualities of DNA amplification or printing. `vsn` cannot be used to simultaneously calibrate and transform data from different batches.

How to reliably diagnose and deal with such violations is beyond the scope of this vignette; see the references for more [5, 3].

Variance. A further assumption that `vsn` makes is that the measurement error (more exactly: the variance) is the sum of two contributions: an additive component that has roughly the same size for all probes on an array, and a multiplicative component that is roughly proportional in size to the signal's true value, with a proportionality factor (called the *coefficient of variation*) that is the same for all genes [4].

Most genes unchanged assumption. `vsn` assumes that only a minority of genes on the arrays is detectably differentially transcribed across the experiments. The allowed size of that minority is controlled by the parameter `lts.quantile` and must be less than or equal to 1/2. By default, `lts.quantile`=0.5, that is, the maximum.

Processing biases. Image analysis software for cDNA arrays typically estimates a *local background* associated with each probe intensity. For Affymetrix arrays, the intensities from mismatch probes are thought to represent the level of non-specific signal. In both cases, the raw probe intensities may be *adjusted* by subtracting these background estimates. Some software packages, however, bias the adjustment through rules based on the data values. For example, Affymetrix' MAS 5.0 software uses the mismatch intensity only if it is smaller than the probe's intensity, and otherwise employs a heuristic to make sure that the net intensities always remain positive. As a consequence, the intensities are systematically over-estimated, and cannot be used with `vsn`. For Affymetrix data, we recommend to use `vsn` on the probe intensities from the "CEL file". For cDNA data, we recommend to use only background adjustment procedures that estimate the background independent of the observed foreground intensity.

[3]It has been reported that for some genes the dye bias is different from gene to gene, such that the proportionality factor does not simply factorize as in (3). As long as this only occurs sporadically, this should not have much effect on the estimation of the calibration and variance stabilization parameters. Further, by using an appropriate experimental design such as color-swap or reference design, the effects of gene-specific dye-biases to subsequent analyses can also be reduced.

References

[1] W. Huber, A. von Heydebreck, H. Sültmann, A. Poustka, and M. Vingron. Variance stablization applied to microarray data calibration and to quantification of differential expression. *Bioinformatics*, 18:S96–S104, 2002.

[2] W. Huber, A. von Heydebreck, H. Sültmann, A. Poustka, and M. Vingron. Parameter estimation for the calibration and variance stabilization of microarray data. *Statistical Applications in Genetics and Molecular Biology*, Vol. 2: No. 1, Article 3, 2003. http://www.bepress.com/sagmb/vol2/iss1/art3

[3] W. Huber, A. von Heydebreck, and M. Vingron. Analysis of microarray gene expression data. To appear in the *Handbook of Statistical Genetics*, 2003. Eds.: D. J. Balding, M. Bishop, C. Cannings. John Wiley & Sons, Inc.,

[4] David M. Rocke and Blythe Durbin. A model for measurement error for gene expression analysis. *Journal of Computational Biology*, 8:557–569, 2001.

[5] S. Dudoit, Y. H. Yang, T. P. Speed, and M. J. Callow. Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. *Statistica Sinica*, 12:111–139, 2002.

[6] J. D. Storey and R. Tibshirani. SAM thresholding and false discovery rates for detecting differential gene expression in DNA microarrays. To appear in: G. Parmigiani, E. S. Garrett, R. A. Irizarry and S. L. Zeger (eds.), The analysis of gene expression data: Methods and software. New York, Springer 2003.

[7] A. A. Alizadeh et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, 2000.