Robust calibration and variance stabilization with VSN

Wolfgang Huber

October 28, 2005

Contents

1 Getting started: brief overview

vsn is a method to preprocess microarray intensity data. Calling *vsn* can be as simple as

```
> e2 <- vsn(e1)
```

where `e1` is an `exprSet` with raw data and `e2` one with calibrated and *glog*-transformed data. `e1` can also be a matrix, a data frame with numeric columns only, or an object of class *marrayRaw*. It contains the raw intensity measurements from the DNA probes on a series of microarrays (with rows corresponding to probes, columns to arrays and/or dyes).

The so-called *glog* (short for ***generalized logarithm***) is a function that is like the natural logarithm for large values (large compared to the background noise), but is less steep for smaller values. Differences between the transformed values are the ***generalized***

log-ratios. These are *shrinkage estimators* of the natural logarithm of the fold change. For example,

```
> M <- exprs(e2)[, i] - exprs(e2)[, j]
```

produces the vector of generalized log-ratios between samples i and j. Notes that these are to base e. If you want to convert them to base 2, divide them by log(2). If you want to know the estimated the fold change, exponentiate:

```
> M.base2 <- M/log(2)
> fold.change <- exp(M)
```

Generalized log-ratios can be viewed as a *shrinkage estimator*: they are always smaller than or equal to the naive log-ratios; equality is asymptotically reached if the probe intensities are large both for samples i and j. Their advantage is that they do not suffer from the variance divergence of the naive log-ratios at small intensities: they remain well-defined and statistically meaningful when the data come close to zero or even become negative. Please consult the references for more on the mathematical-methodical background [1, 2, 3].

In short, each column is calibrated by an affine transformation[1], then the whole data are 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. Many statistical methods such as hypothesis tests, ANOVA modeling, clustering, or classification work better or are easier to use if the variance of the data is roughly the same for all observations[2].

2 Running vsn on data from a single two-color array

The package includes example data from a cDNA array on which two biologically very 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.

```
> nkid <- vsn(kidney)
```

[1] It is possible to stratify the transformations within columns; this is discussed in Section **??**

[2] 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))
> log.na = function(x) log(ifelse(x > 0, x, NA))
> plot(exprs(nkid), main = "vsn", pch = ".")
> plot(log.na(exprs(kidney)), main = "raw", pch = ".")
```

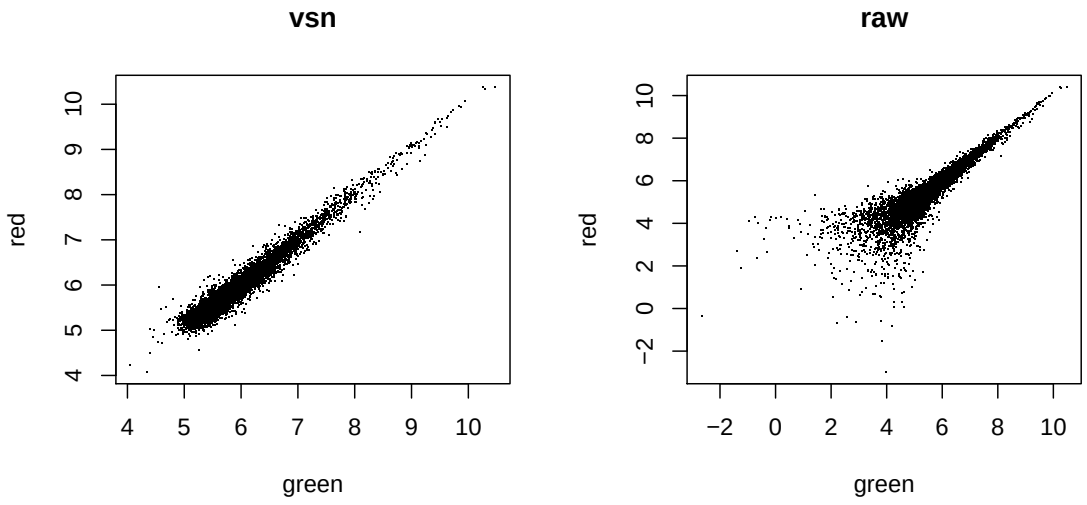

Figure 1: Scatterplots of the kidney example data

`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[3]. Within each window, the median may be considered a pooled

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

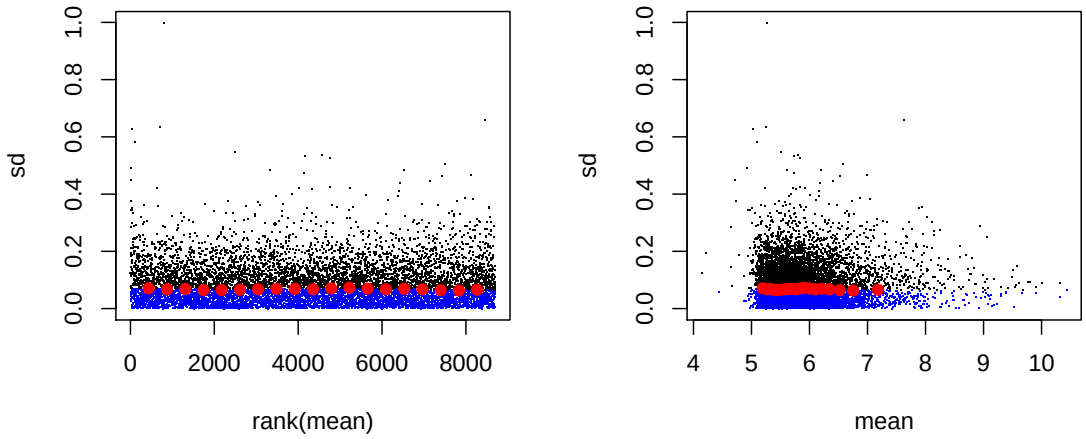

Figure 2: Standard deviation versus rank of the mean, and the mean, respectively

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 a horizontal line. It may have some random fluctuations, but should not show an overall trend. If this is not the case, that usually indicates a data quality problem, or is a consequence of inadequate prior data preprocessing (see Section 8). The rank ordering distributes the data evenly along the x-axis. 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 before the last iteration. If this is not the case, the number of iterations may be increased through the parameter `iter`. It could also indicate a data quality problem, see Section 8.

The generalized log-ratios for this experiment can be obtained for further processing through

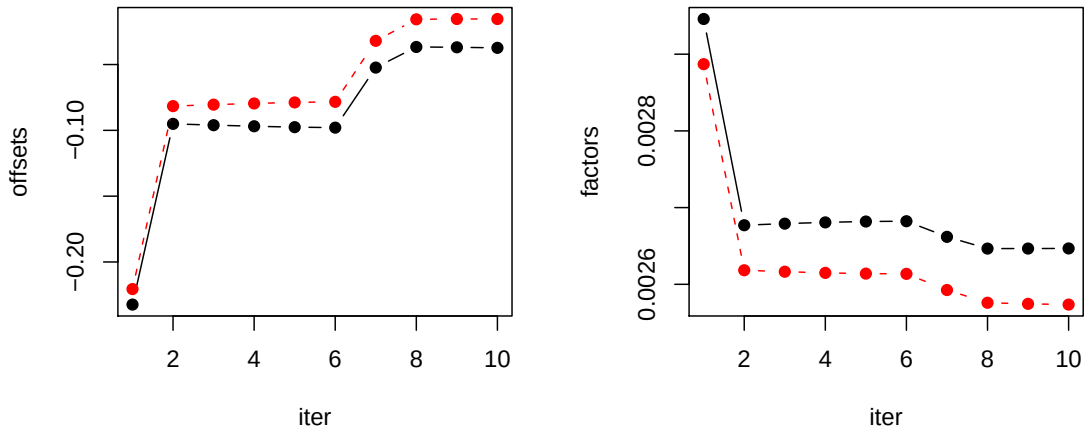

Figure 3: Iteration trajectory of the calibration and transformation parameters

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

The histogram is shown in Fig. 3.

3 Running vsn on data from multiple arrays ("single color normalization")

The package includes example data from a series of 8 cDNA arrays on which different lymphoma were hybridized together with a reference cDNA [6].

```
> data(lymphoma)
> dim(exprs(lymphoma))

[1] 9216   16

> pData(lymphoma)

      name    sample
1  lc7b047 reference
2  lc7b047    CLL-13
```

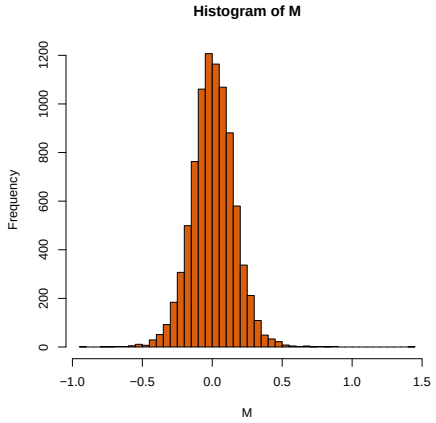

Figure 4: Histogram of generalized log-ratios for the kidney example data

```
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 $2, 4, \ldots, 16$. We can call `vsn` on all of them at once:

```
> lym <- vsn(lymphoma)
```

This calculation may take a while.

```
> meanSdPlot(lym)
```

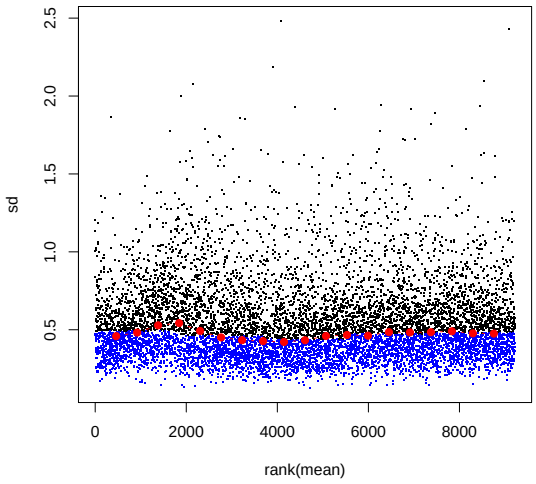

Figure 5: Standard deviation versus rank of the mean for the lymphoma example data

Again, Fig. 5 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. 6 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).

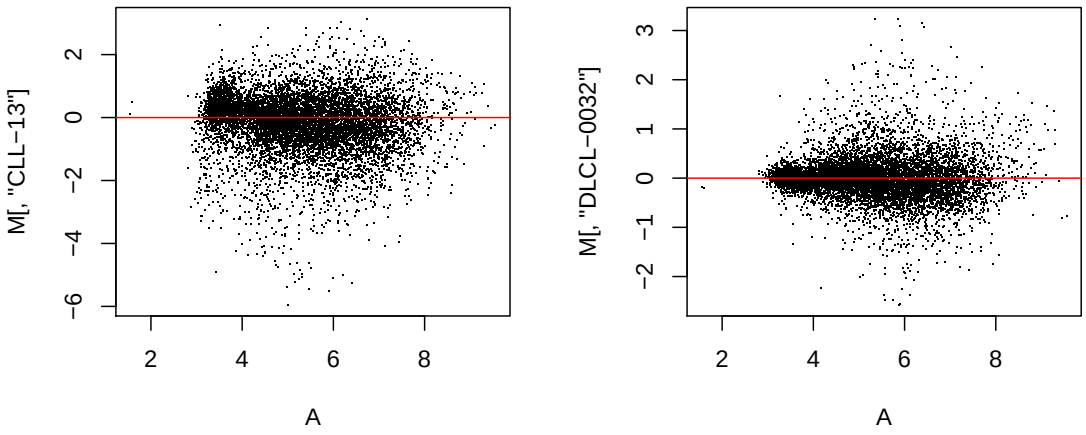

Figure 6: Mean-difference plots for two slides from the lymphoma example data

4 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 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. Affymetrix genechips preprocessing involves the following steps: (i) combining the perfect match (PM) and mismatch (MM) intensities into one number per probe, (ii) calibrating, (iii) transforming, and (iv) summarizing. `vsn` addresses the calibration and transformation. We can use the function `expresso` to run the whole process in one go:

```
> 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")
```

Here, we have ignored the MM values and used `medianpolish` for summarization, as in the RMA method [8]. For comparison, let's calculate expression values using another

normalization method. The resulting plots are shown in Fig. 7.

```
> es2 = expresso(Dilution[, 1:2], bgcorrect.method = "rma", normalize.method = "quantiles",
+     pmcorrect.method = "pmonly", summary.method = "medianpolish")
> x1 = exprs(es1)
> x2 = exprs(es2)

> 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")
```

Note that while the values of *vsn* are normally represented on the natural logarithmic scale, with the wrapper `normalize.AffyBatch.vsn` they are transformed to the logarithm base 2 scale. This way `normalize.AffyBatch.vsn` fits into the conventions of the *expresso*-function.

5 Running vsn from the limma package

`vsn` can be called from the limma package through the function `normalizeBetweenArrays`:

```
> library(limma)
> MA <- normalizeBetweenArrays(RG, method = "vsn")
```

Note that `RG` should contain raw intensities, i. e., prior log-transformation or any normalization. The returned intensities and log-ratios in `MA` are on the $\log -2$ scale, not the $\log -e$ scale as when `vsn` is called directly. Please see also the help page for `normalizeBetweenArrays`.

For print-tip wise normalization, construct a function `pinId` that calculates the print-tip ID (1...16) for every spot:

```
> pinId <- function(x) unlist(lapply(1:(x$ngrid.r * x$ngrid.c),
+     rep, x$nspot.r * x$nspot.c))
```

and call

```
> MA <- normalizeBetweenArrays(RG, method = "vsn", strata = pinId(thelayout))
```

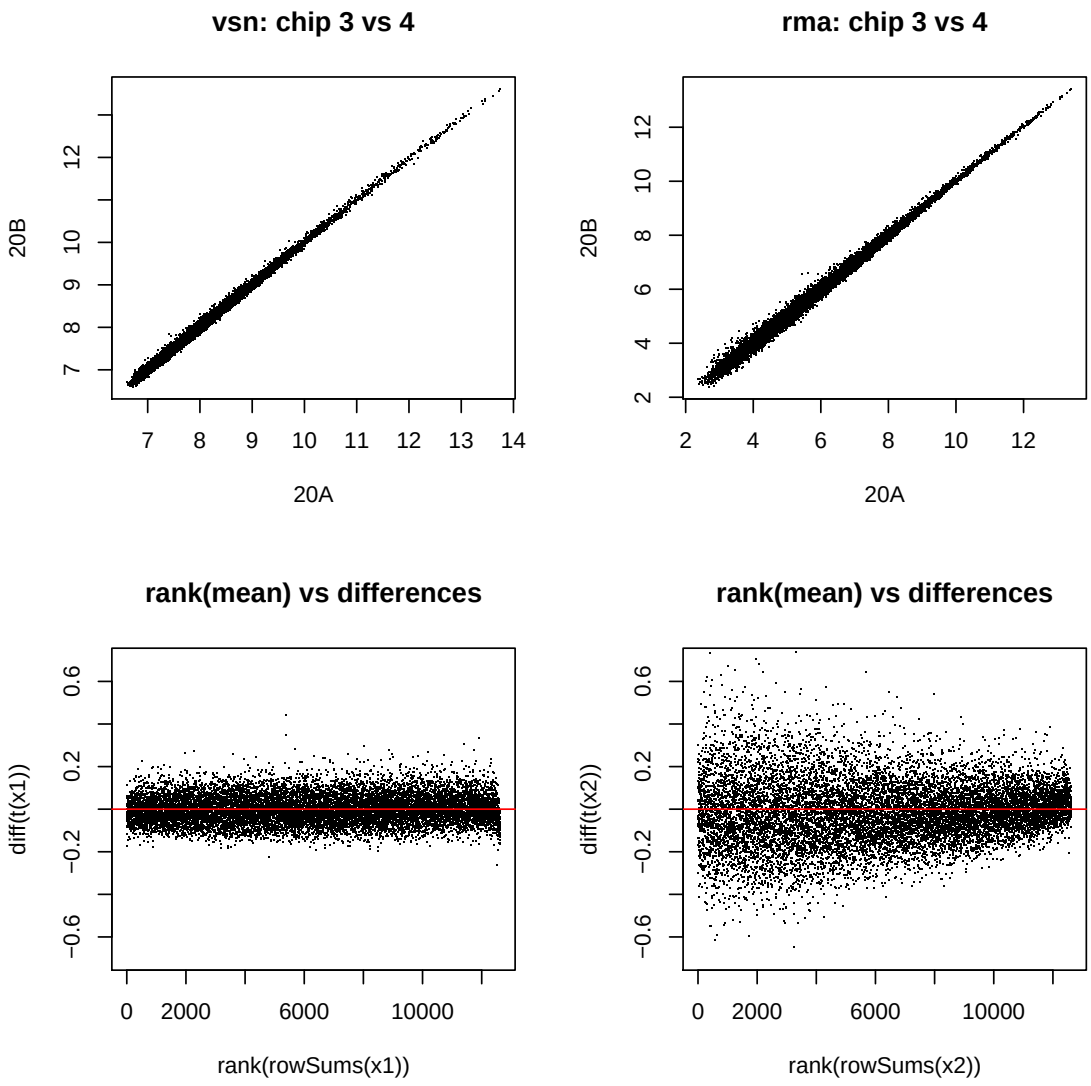

Figure 7: `normalize.AffyBatch.vsn` example

6 The calibration parameters

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 λ_{si} and shifting by o_{si}:

$$y'_{ki} = \frac{y_{ki} - o_{si}}{\lambda_{si}}, \tag{2}$$

where $s \equiv s(k)$ is the so-called *stratum* for probe k. In the simplest case, there is only one stratum, i. e. the index s is always equal to 1, or may be omitted altogether. This amounts to assuming that that the data of all probes on an array were subject to the same systematic effects, such that an array-wide calibration is sufficient.

A model with multiple strata per array may be useful for spotted arrays. For these, stratification may be according to print-tip [5] or PCR-plate [2]. For oligonucleotide arrays, it may be useful to stratify the probes by physico-chemical properties, e. g. to assume that probes of different sequence composition attract systematically different levels of unspecific background signal.

The transformation to a scale where the variance of the data is approximately independent of the mean is

$$h_{ki} = \operatorname{arsinh}(a_0 + b_0 y'_{ki}) = \log\left(a_0 + b_0 y'_{ki} + \sqrt{(a_0 + b_0 y'_{ki})^2 + 1}\right). \tag{3}$$

Eqns. (2) and (3) can be combined, so that the whole transformation is given by

$$h_{ki} = \operatorname{arsinh}(a_{si} + b_{si} y_{ki}). \tag{4}$$

Here, $a_{si} = a_0 - b_0 o_{si}/\lambda_{si}$ and $b_{si} = b_0/\lambda_{si}$ are the combined calibation and transformation parameters for probes from stratum s and sample i.

We can access the calibration and transformation parameters through

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

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

> prep$vsnParams

, , 1

          [,1]        [,2]
[1,] -0.037301 -0.01540317

, , 2

           [,1]        [,2]
[1,] 0.00264677 0.002573589
```

The `description` slot of an *exprSet* is an object of class *MIAME*, and may contain annotation data pertinent to the experiment represented by the object. For an exprSet with d sample and n_s probe strata (see Section **??**), `prep$vsnParams` is a numeric array with dimensions $(n_s, d, 2)$. `prep$vsnParams[s, i, 1]` is what was called a_{si} in Eqn. (4), and `prep$vsnParams[s, i, 2]` is b_{si}. Compare the numbers printed above with the final values in Fig. 3.

6.1 More on calibration

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[1, , ]

            [,1]        [,2]
[1,] -0.07043781 0.002663749
[2,] -1.34136767 0.010349088
```

The factor for the red channel is now about four times as large as before. The result is shown in Fig. 8.

7 Assessing vsn

The function `vsn` is a parameter estimation algorithm that fits the parameters for a certain model. In order to see how good the estimator is, we can look at bias, variance, sample size dependence, robustness against model misspecificaton and outliers. This is done in the document `convergence.pdf`, which can be found in the `doc` subdirectory of the package.

Practically, the more interesting question is how different microarray calibration and data transformation methods compare to each other. For this, one needs to specify a measure of goodness. One approach is to compare the obtained values against a known truth. This 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

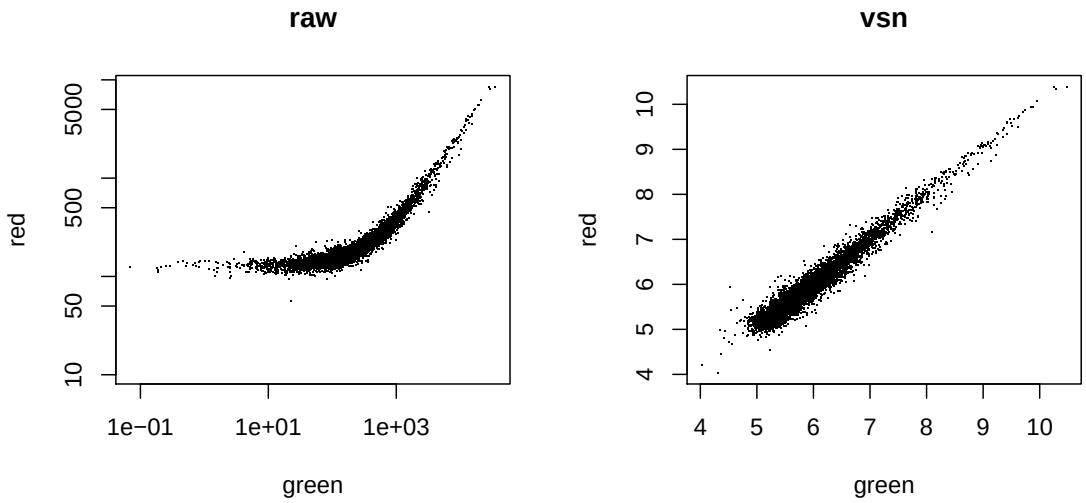

Figure 8: Scatterplots for badly biased data. Left hand side: raw data on log-log scale, right hand side: after calibration and transformation with vsn.

relevant spike-in concentrations. Like any statistical method, one can make different choices with respect to the trade-off between bias and variance [7].

Here, we focus on one particular aspect: the overall sensitivity and specificity in detecting differential transcription. The following type of analysis can be applied to any data set that contains replicated measurements made on samples from biologically distinct, known groups. The idea is that we compare the within-group variability (among the biological replicates) to the between-group variability (between different tissue types). The smaller the former and the larger the latter, the better we may deem the performance of the calibration and data transformation.

Here, as a measure of the relative size of between- and within-group variability we take the size of the t-statistics. The acceptable use of CPU time and disk memory of a package vignette is limited, thus here we stick to a very simple-minded calculation, and a small data set. See Fig. 9 and the calculations below. Two applications to larger data sets are described in reference [1]. More sophisticated analyses can be made by comparing not just the distributions of t-statistics, but for example, the estimated false discovery rates, using different test statistics. You are encouraged to try this out with your own data.

```
> library(marray)
> mr <- new("marrayRaw", maGf = exprs(lymphoma)[, refrs], maRf = exprs(lymphoma)[,
+     samps], maLayout = new("marrayLayout", maNgr = 4, maNgc = 4,
```

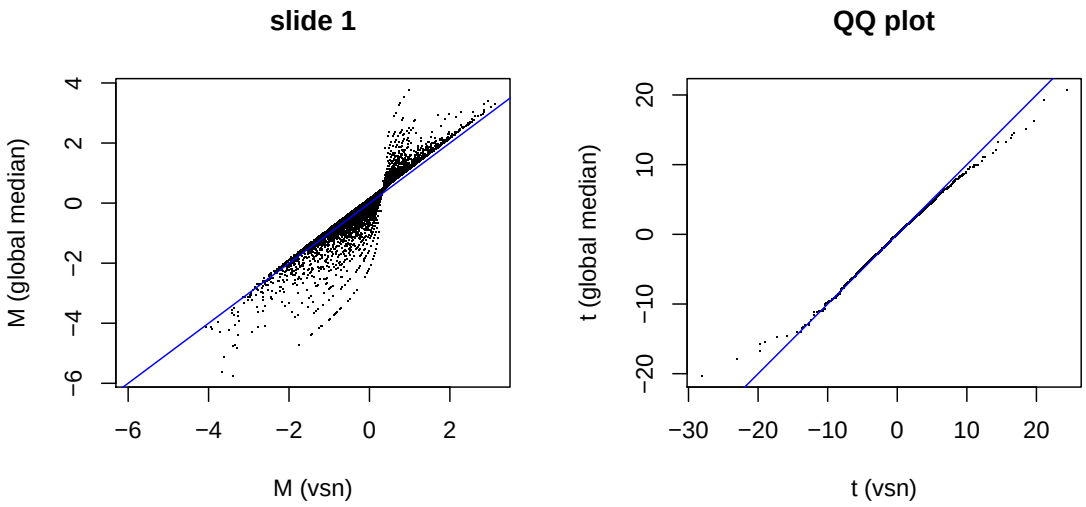

Figure 9: Left hand side: x-axis – generalized log-ratios for slide 1 from `vsn`, y-axis – log-ratios for slide 1 after global median normalization. For most genes, the two are the same, but in some cases the generalized log-ratio is smaller. It is never larger. This demonstrates the *ratio shrinkage* by the variance stabilization. Right hand side: quantile-quantile-plot of the t-statistic for the comparison between the 4 slides with CLL and the 4 with DLCL. The t-statistics from `vsn` are larger, i. e. it compares more favorable with respect to the relative size of between- and within-group variability.

```
+     maNsr = 24, maNsc = 24))
> mn <- maNorm(mr, norm = "median", echo = TRUE)

> par(mfrow = c(1, 2))
> library(multtest)
> plot(M[, 1], mn@maM[, 1] * log(2), xlab = "M (vsn)", ylab = "M (global median)",
+     main = "slide 1", pch = ".")
> abline(a = 0, b = 1, col = "blue")
> classlabel <- regexpr("CLL", colnames(M)) > 0
> t1 <- mt.teststat(M, classlabel)
> t2 <- mt.teststat(mn@maM, classlabel)
> qqplot(t1, t2, xlab = "t (vsn)", ylab = "t (global median)",
+     main = "QQ plot", pch = ".")
> abline(a = 0, b = 1, col = "blue")
```

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:

$$y_{ik} \approx a_i + b_i b_k c_{ik}. \tag{5}$$

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[4].

Systematic effects associated with print-tip, PCR, or probe-sequence Equation 5 can be generalized to

$$y_{ik} \approx a_{is} + b_{is} b_k c_{ik}. \tag{6}$$

[4]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 (5). 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.

that is, the background term a_{is} and the gain factor b_{is} can be different for different groups s of probes on an array. For example, with cDNA microarray data, it could be advantageous to fit different parameters for each print-tip group of spots, or for groups of spots whose DNA was PCR–amplified and stored in the same microtitre plate. For Affymetrix chips, one can find systematic dependences of the affinities b_{is} or the background terms a_{is} on the probe sequence. This can be addressed by using the `strata` argument of the function `vsn`.

Situations in which the assumptions (5) or (6) are 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.

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, 2].

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 (less than half) of genes on the arrays is detectably differentially transcribed across the experiments. If it is safe to assume that a smaller fraction of genes is non-negligibly differentially transcribed, the efficiency of the estimation can be improved by increasing the parameter `lts.quantile` from its default value of 0.5 to a value between 0.5 and 1.

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.

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, 13

[2] 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., 2, 11, 16

[3] 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 2

[4] David M. Rocke and Blythe Durbin. A model for measurement error for gene expression analysis. *Journal of Computational Biology*, 8:557–569, 2001. 16

[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. 7, 11, 16

[6] A. A. Alizadeh et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403:503–511, 2000. 5

[7] L. M. Cope, R. A. Irizarry, H. A. Jaffee, Z. Wu, and T. P. Speed. A Benchmark for Affymetrix GeneChip Expression Measures. *Bioinformatics*, 20:323–331, 2004. 13

[8] R. A. Irizarry, B. Hobbs, F. Collin, Y. D. Beazer-Barclay, K. J. Antonellis, U. Scherf, and T. P. Speed. Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics* 4:249–264, 2003. 8