Assessing the accuracy of vsn with simulated data

Wolfgang Huber

February 11, 2007

Contents

1 Overview

The purpose of this vignette is to assess that the software in *vsn* actually does what it is supposed to do according to the mathematical theory. And to see *how fast* (or slow, depending on your point of view) and *how accurately* it does that.

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*. This vignette demonstrates some examples. Please refer to reference [1] for more detail on the simulation model, the assessment strategy and a comprehensive suite of assessments with respect to the number of features `n`, the number of arrays `d`, the fraction of differentially expressed genes `de`, and the fraction of up-regulated genes `up`.

2 Number of features n

Fig. 1 shows the estimation error for the transformation (i. e. the root mean squared difference between true and estimated transformed data) as a function of the number of features n. If `vsn` works correctly, the estimation error should decrease roughly as $n^{-1/2}$.

```
> makeFig("fign1", 5, 5, {
+     n = 1000 * 2^seq(-2, 5)
```

```
+     res = sim(n = n)
+     myPlot(n, res, main = "n: one stratum")
+ })

> makeFig("fign2", 5, 5, {
+     res = sim(n = n, nrstrata = 8)
+     myPlot(n, res, main = "n: 8 strata")
+ })
```

3 Number of samples d

Fig. 2a shows the estimation error as a function of the number of samples d. This curve is essentially flat. This is because the number of parameters that need to be estimated is proportional to d, so the "number of data points per parameter" is constant in this plot (in contrast to Fig. 1).

```
> makeFig("figd", 5, 5, {
+     d = 2^seq(1, 5)
+     res = sim(d = d)
+     myPlot(d, res, main = "d")
+ })

null device
          1
```

4 Number of strata

In Fig. 2b, we see the estimation error as a function of the number of strata. It should increase, since for each stratum, we need to estimate separate parameters, and if the overall number of features does not change, more strata means less and less data per parameters.

```
> makeFig("fignrstrata", 5, 5, {
+     nrstrata = 2^seq(0, 4)
+     res = sim(nrstrata = nrstrata)
+     myPlot(nrstrata, res, main = "nrstrata")
+ })

null device
          1
```

5 Differentially expressed genes

In the following code, `de` is the fraction of differentially expressed genes. We run the simulation both with default settings `lts.quantile=0.9` and the more robust `lts.quantile=0.5`. The reason why `lts.quantile=0.5` is not the default is that the estimator is more efficient (more precise with less data) *if* the fraction of differentially expressed genes is not that large. See Figure 3.

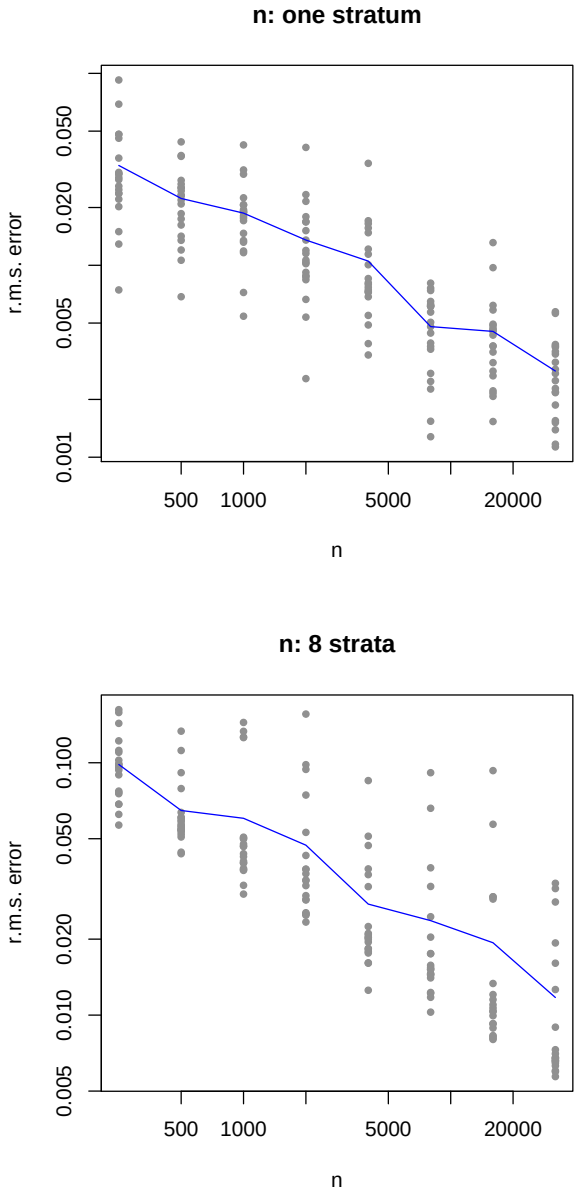

Figure 1: Estimation error as a function of the number of features n. If `vsn` works correctly, the estimation error should decrease roughly as $n^{-1/2}$.

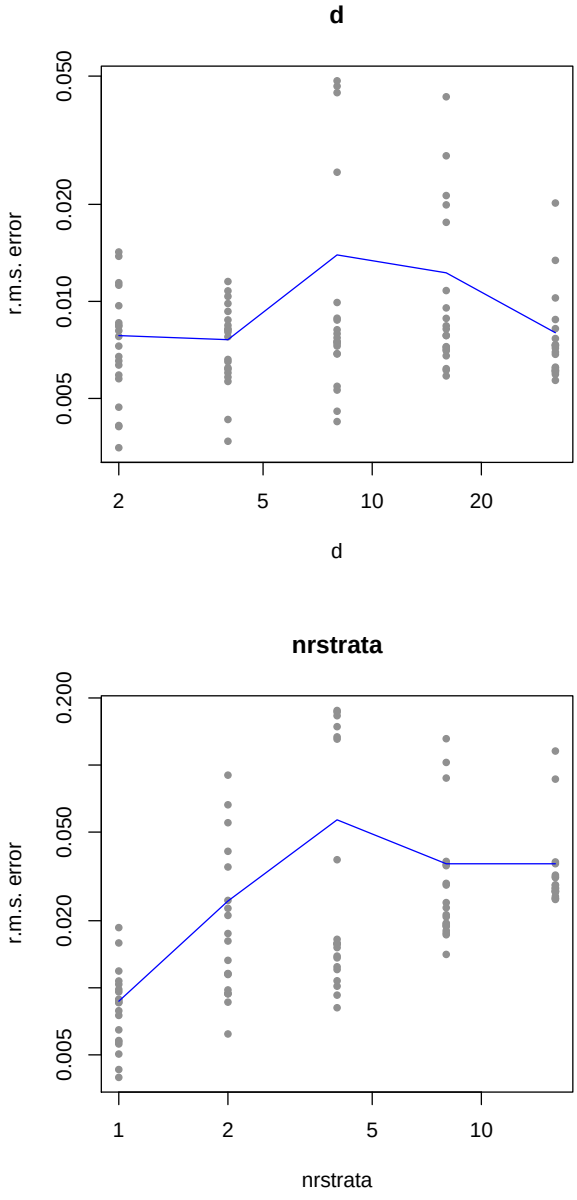

Figure 2: Estimation error as a function of (a) the number of samples and (b) the number of strata. See Sections 3 and 4.

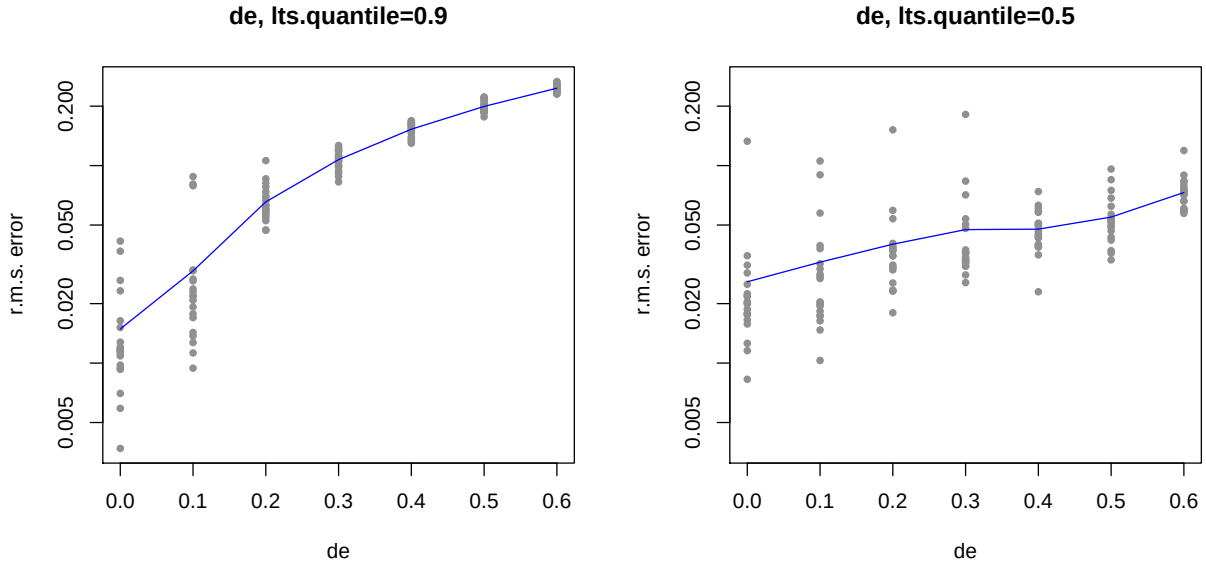

Figure 3: Estimation error as a function of the number of differentially expressed genes, for two different settings of `lts.quantile`; see Section 5.

```
> makeFig("figdiff", 10, 5, {
+     de = (0:6)/10
+     res1 = sim(de = de, nrstrata = 2)
+     res2 = sim(de = de, nrstrata = 2, lts.quantile = 0.5)
+     myPlot2(de, list("de, lts.quantile=0.9" = res1, "de, lts.quantile=0.5" = res2))
+ })
```

```
null device
          1
```

up is the fraction of up-regulated genes among the differentially expressed genes. The best results are obtained for up≈ 0.5, while the estimation goes up the more unbalanced the situation becomes.

```
> makeFig("figup", 10, 5, {
+     up = (0:8)/8
+     res1 = sim(up = up, nrstrata = 2, de = 0.2)
+     res2 = sim(up = up, nrstrata = 2, de = 0.2, lts.quantile = 0.5)
+     myPlot2(up, list("up, lts.quantile=0.9" = res1, "up, lts.quantile=0.5" = res2))
+ })
```

```
null device
          1
```

6 Missing values

```
> makeFig("figmiss1", 5, 5, {
+     miss = seq(0, 0.75, length = 7)
```

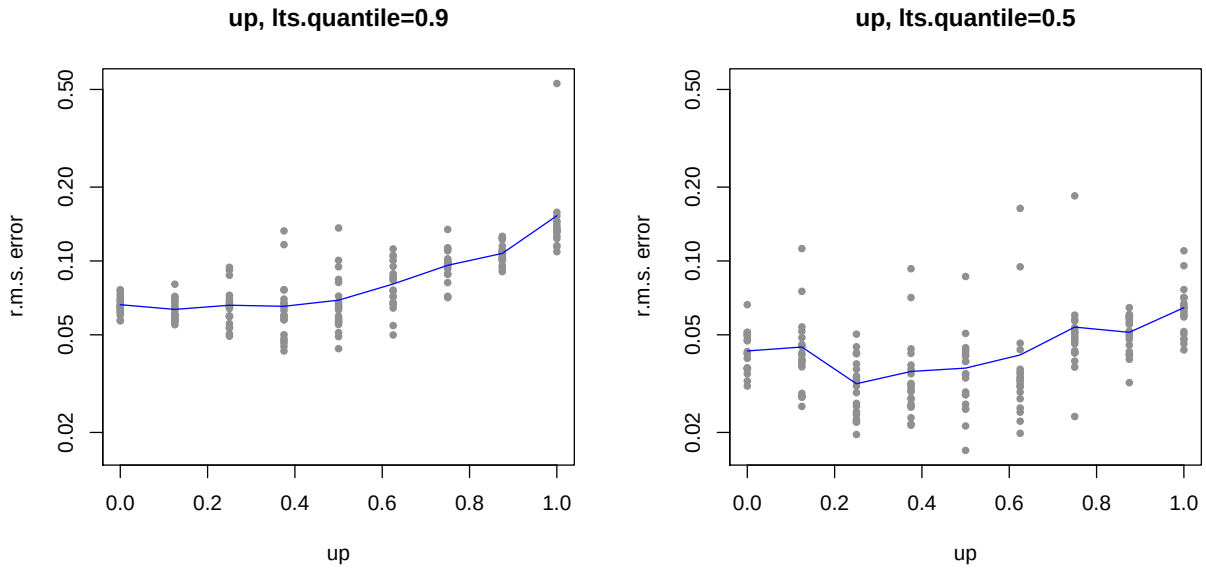

Figure 4: Estimation error as a function of the fraction of up-regulated genes. for two different settings of `lts.quantile`; see Section 5.

```
+     res = sim(miss = miss, d = 8)
+     myPlot(miss, res, log = "y", main = "fraction NA (d=8)")
+ })

null device
          1

> makeFig("figmiss2", 5, 5, {
+     miss = seq(0, 0.1, length = 5)
+     res = sim(miss = miss, d = 2)
+     myPlot(miss, res, log = "y", main = "fraction NA (d=2)")
+ })

null device
          1
```

7 Incremental normalization

```
> dat = sagmbSimulateData(n = 10000, d = 12, de = 0, nrstrata = 1,
+     miss = 0, log2scale = TRUE)
> v = new("vsn", refh = dat$mu, refsigma = dat$sigma, n = length(dat$mu))
> fit = vsn2(dat$y, lts.quantile = 1, verbose = FALSE)
> parRef = array(as.numeric(NA), dim = dim(fit@par))
> for (j in 1:ncol(dat$y)) {
+     vj = vsn2(dat$y[, j], reference = v, lts.quantile = 1, verbose = FALSE)
```

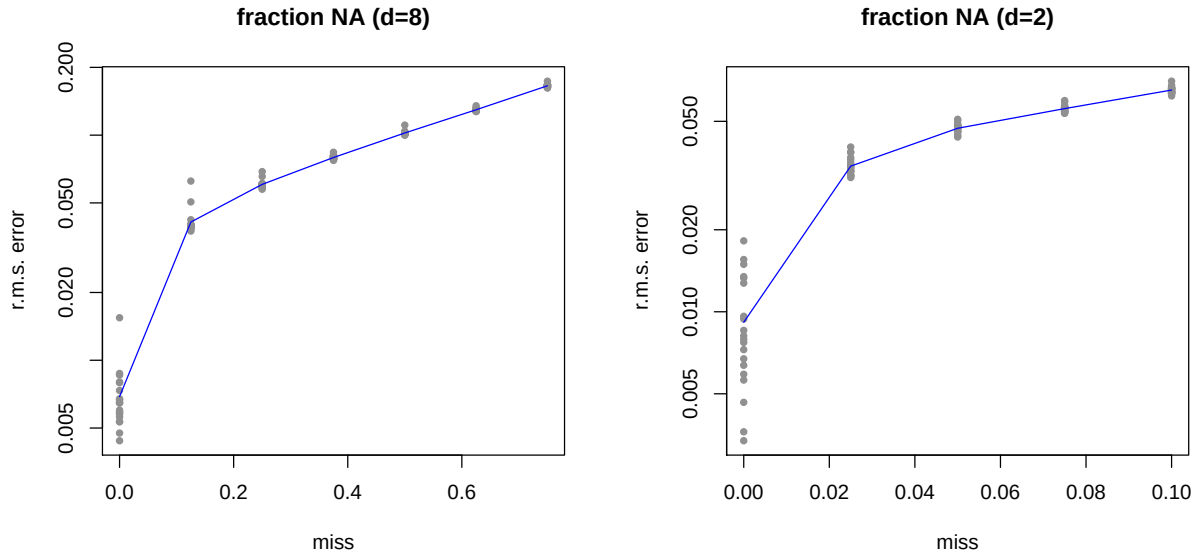

Figure 5: Estimation error as a function of the fraction of missing data points, see Section 6.

```
+     parRef[, j, ] = vj@par
+ }
> makeFig("figincr", 10, 10, {
+     par(mfcol = c(2, 2))
+     for (k in 1:2) {
+         plot(dat$par[1, , k], parRef[1, , k], xlab = "True",
+             ylab = "Reference fit", main = c("offset", "factor")[k])
+         abline(a = 0, b = 1)
+         plot(fit@par[1, , k], parRef[1, , k], xlab = "Profile Likelihood fit",
+             ylab = "Reference fit", main = c("offset", "factor")[k])
+         abline(a = 0, b = 1)
+     }
+ })
null device
          1
```

8 Subsampling

TODO: Some sort of test that checks for correct functioning of the subsampling if there are multiple strata.

References

[1] W. Huber, A. von Heydebreck, H. Sültmann, A. Poustka, and M. Vingron. Parameter estimation for the calibration and variance stabilization of microarray data. *Sta-*

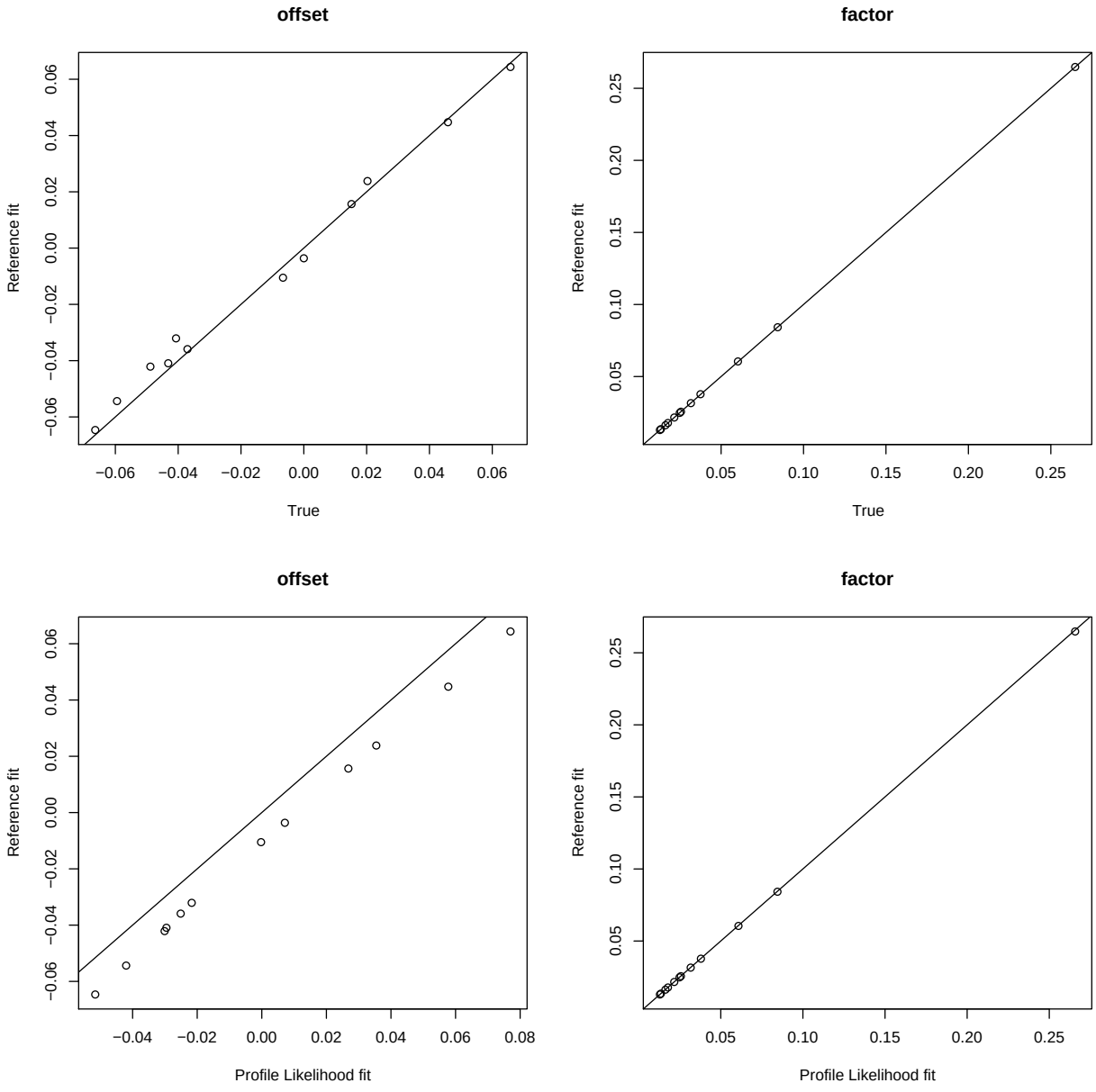

Figure 6: Comparison of parameters fitted from incremental normalisation (y-axis) with true parameters (x-axis, upper row) and with parameters fitted from joint profile-likelihood normalisation (x-axis, lower row); see Section 7.

tistical Applications in Genetics and Molecular Biology, Vol. 2: No. 1, Article 3, 2003. http://www.bepress.com/sagmb/vol2/iss1/art3