Bioconductor's nnNorm package

Laurentiu A. Tarca[1]

September 14, 2004

1. CRBF, Laval University, CANADA, http://www.crbf.ulaval.ca

Contents

1 Overview

The nnNorm package contains mainly a function for intensity and spatial normalization of cDNA two color data, or paired single chanel data, based on neural networks fitting. Functionality to compare the distributions of the normalized log ratios is also provided. For the simpler case when only intensity normalization is performed (univariate distortion color model), we provide functionality to store, and use the models obtained for each print tip of each slide.

Introduction to intensity and spatial normalization. This document provides a tutorial for using the nnNorm package. For a detailed description of the principles and algorithmics implemented by this package consult Tarca et al. (2004).

Spatial and intensity normalization implemented in nnNorm. The nnNorm package implements intenisty and spatial normalization of cDNA data based on neural networks. In this apporach the log average intensity A as well as the spatial coordinates X and Y (obtained by binning the print tip group space) are used as regressors into a neural network model. Resistence to outlieres is enhanced by assigning to each spot a weigth which is dependent on how extreme its log ratio, M, is with respect to that of the spots having about the same A level belonging to the same print tip group.

Help files. As with any R package, detailed information on functions, their arguments and value, can be obtained in the help files. For instance, to view the help file for the function maNormNN in a browser, use help.start() followed by ? maNormNN.

2 Case study: Apo AI dataset, Callow et al.

We demonstrate the functionality of this package using gene expression data from the Apo AI study of Callow et al. (2000). To load the Apo dataset in a object called Apo of type marrayRaw, we use the following lines:

```
> library(marray)
> dataweb <- "http://www.stat.berkeley.edu/users/terry/zarray/Data/ApoA1/rg_a1ko_morph.txt"
> data <- read.table(dataweb, header = TRUE, sep = ",", dec = ".")
> ApoLayout <- new("marrayLayout", maNgr = 4, maNgc = 4, maNsr = 19,
+     maNsc = 21)
> spots <- 4 * 4 * 19 * 21
> Gb <- Rb <- array(c(0), c(spots, 16))
> Gf <- as.matrix(data[, seq(2, 33, 2)])
> Rf <- as.matrix(data[, seq(3, 33, 2)])
> labs <- c(paste(c("c"), 1:8, sep = ""), paste(c("k"), 1:8, sep = ""))
> maInfo <- new("marrayInfo", maLabels = labs, maInfo = data.frame(Names = labs))
> Apo <- new("marrayRaw", maRf = Rf, maGf = Gf, maRb = Rb, maGb = Gb,
+     maLayout = ApoLayout, maTargets = maInfo)
```

Note that the background values were set to zero as the data file contains background corected red and green intensities.

Due to several factors, the bias in the microarray experiments may depend not only on the average log intensity level but also on the spatial coordinates Yang et al. (2002). Figure 1 shows the reconstructed image of log ratios `M`, for slide No. 16 in the Apo AI data set. This image is obtained using the `maImage` function of `marray` package.

```
> RGcol <- maPalette(low = "green", high = "red", k = 20)
> maImage(Apo[, 16], x = "maM", col = RGcol, zlim = c(-2, 2))
```

Observe the clearly uneven distribution of M values in the print tip groups situated at the bottom of the figure.

Now we perform normalization with the method of the `nnNorm` package called `maNormNN`. This function returns a list with two components: `batchn`, a `marrayNorm` type object (containing the normalized log ratios) and `models` containing the parameters of the models and ranges of validity. There is a distinct model for every print tip group aside in every slide. The `maNormNN` function is therefore a print tip oriented normalization method.

```
> library(nnNorm)
> set.seed(20)
> ApNN2DA <- maNormNN(Apo)$batchn
```

Figure 2 shows the log ratios, for the same data as in Figure 1 after intensity and spatial normalization. Observe a rather uniform distribution of red and green colored spots.

```
> maImage(ApNN2DA[, 16], x = "maM", col = RGcol, zlim = c(-2, 2))
```

We may compare the robust neural networks normalization method with other existing approches that takes into account intensity and spatial bias fluctuations, like for e.g. a compozite normalization method avaliable in `marray` package.

```
> AcPLo2D <- maNormMain(Apo, f.loc = list(maNorm2D(), maNormLoess(x = "maA",
+     y = "maM", z = "maPrintTip")), a.loc = maCompNormEq())
```

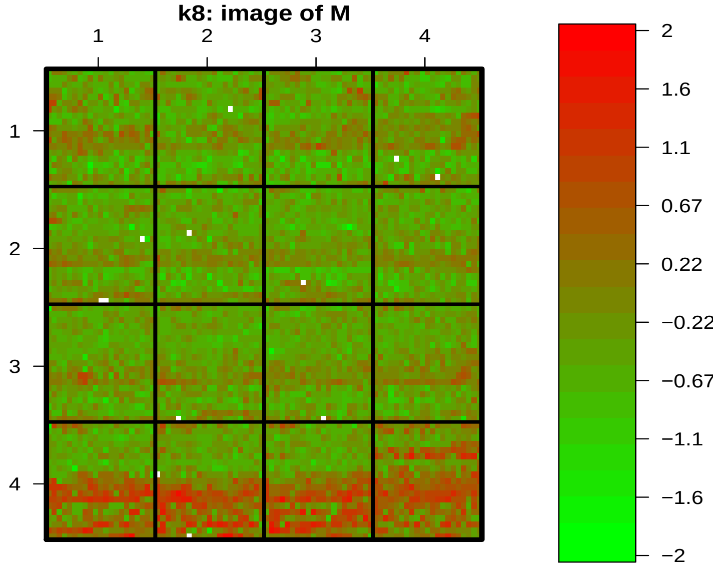

Figure 1: Image of raw M values for Apo data set, slide No. 16.

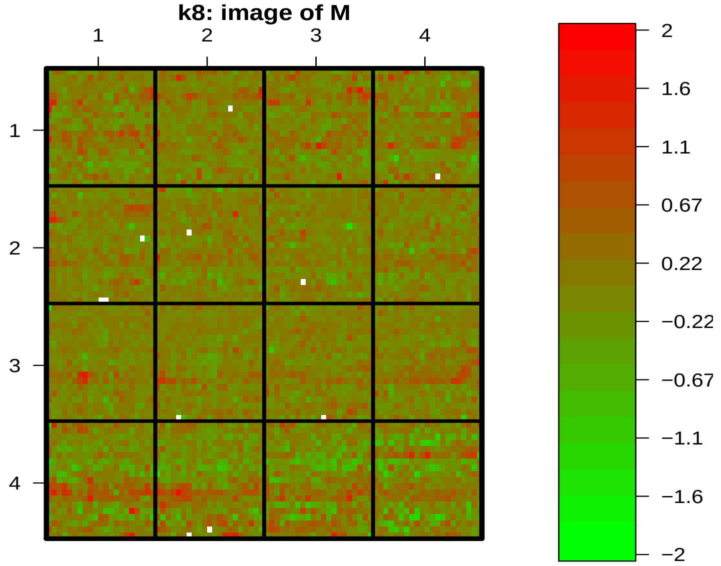

Figure 2: Image of intensity and spatial normalized M values, for Apo data set, slide No. 16.

We define a function able to compute the Westfall and Young (1993) adjusted p-values for different normalizations of the same Apo batch. For this we use functionality of the `multtest` package (Dudoit et al. (2002)). To reduce the computation time we limit the number of permutations to `B=5000`.

```
> library(multtest)
> getP <- function(maObj) {
+     X <- maM(maObj)
+     class <- c(rep(0, 8), rep(1, 8))
+     resT <- mt.maxT(X, class, B = 5000)
+     ord <- order(resT$index)
+     resT$adjp[ord]
+ }
```

Now we get the adjusted p-values for the three situations: raw data (Anone), compozite normalization (cpLo2DA) and robust neural networks based (pNN2DA).

```
> pAnone <- log10(getP(Apo))
> pAcPLo2D <- log10(getP(AcPLo2D))
> pApNN2DA <- log10(getP(ApNN2DA))
```

We plot now the log of adjusted p-values obtained using the normalized data. Results are shown in Figure 3.

```
> kout <- c(2149, 4139, 5356, 540, 1739, 1496, 2537, 4941)
> methods <- c("none", "cPLo2D", "pNN2DA")
> plot(rep(1, length(pAnone)), pAnone, xlim = c(1, 3), ylim = c(-4,
+     0), xlab = "Method", ylab = "log10(p)", axes = FALSE)
> points(rep(1, 8), pAnone[kout], col = "red", pch = 20)
> points(rep(2, length(pAcPLo2D)), pAcPLo2D)
> points(rep(2, 8), pAcPLo2D[kout], col = "red", pch = 20)
> points(rep(3, length(pApNN2DA)), pApNN2DA)
> points(rep(3, 8), pApNN2DA[kout], col = "red", pch = 20)
> axis(1, 1:3, methods)
> axis(2)
> box()
```

The spots with the indices `kout` above are those with smallest t-statistics after loess print tip normalization as in Yang et al. (2001).

Observe in Figure 3 that the 8 genes (represented by filled red circles) may be confidentlly distiguished from the bulk of non-differentially expressed genes at a treshold of p=0.01. As some of the genes may have received about the same adjusted p-value, less that 8 filled circles may be observed for a particular method. Setting the `B` parameter to a larger value enhances the differences between the log p-values.

Now we may want to compare the ability of this new normalization method to reduce the variability of log ratios within slides. A easy way to do that is to look at the distributions of the normalized `M` values. The method `compNorm` of `nnNorm` package will do this (see Figure 4).

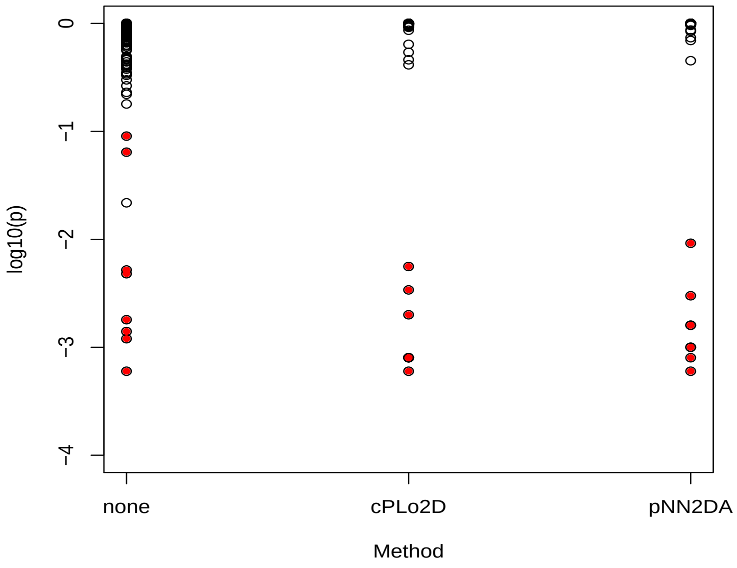

Figure 3: Adjusted p values for raw data (none), composite normalization (cpLo2D), and robust neural networks (pNN2DA)

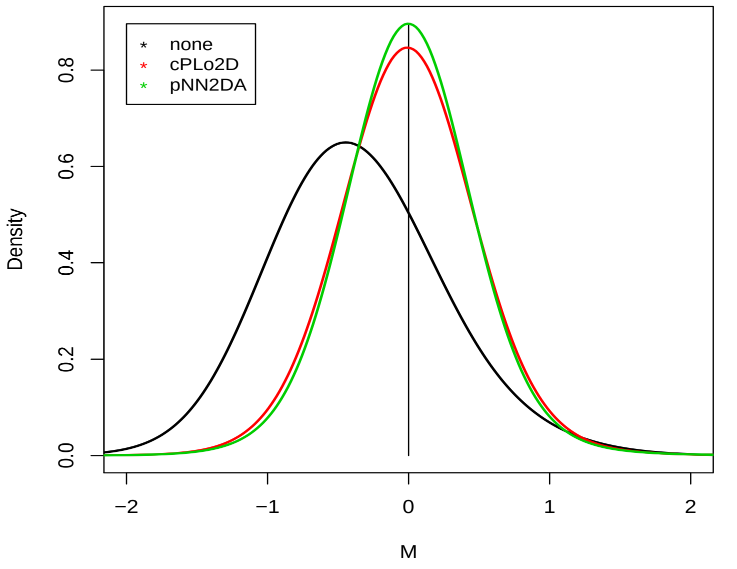

Figure 4: Density plots of normalized log ratios. The highes density for near zero M values coresponds to the robust neural networks mormalization.

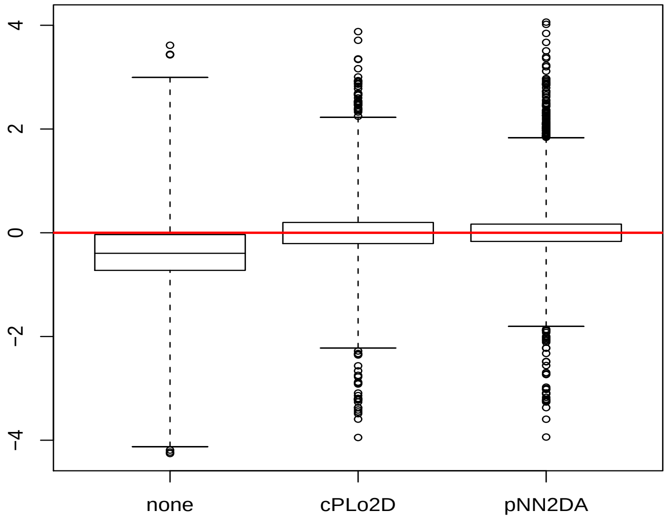

Figure 5: Box plots of normalized log ratios

```
> compNorm(maM(Apo), maM(AcPLo2D), maM(ApNN2DA), bw = 0.9, xlim = c(-2,
+     2), titles = methods, type = "d")
```

The same distributions may be inspected using the box plot in Figure 5.

```
> compNorm(maM(Apo), maM(AcPLo2D), maM(ApNN2DA), bw = 0.9, xlim = c(-2,
+     2), titles = methods, type = "b")
```

The maNormNN function in nnNorm package offers flexibility in choosing the type of dependence of the bias to be accounted for. For e.g. setting binHeight parameter as equal with the number of spot rows in a print tip group, will discard the variable Y in the bias model.

```
> ApoNNy <- maNormNN(Apo[, 1], binWidth = 3, binHeight = maNsr(Apo))$batchn
```

In this case only A and X will be used as regressors for the bias. Similarilly by setting binWidth=maNsc(Apo) in the same function will supress the contribution of the X valiable form the bias estimation. The M-A plots may be shown before and after the normalization for each slide if the parameter maplots is set to "TRUE".

For the case when only intensity normalization is performed (X, and Y regressors discarded) we provide the facility to store and display the unidimensional neural network models. The models parameters and ranges of aplicability needed to make interpolations are stored in the models component of the list returned by the maNormNN function. However, they are not to be manipulated directly by the user. The function predictBias was concieved for this purpose. For e.g. lets perform intensity normalization only for the 13th slide of the Apo batch enabling and disabling the option robust and then compare the resulting estimates:

```
> s <- 13
> pT <- 14
> ApoN <- maNormNN(Apo[, 13], binWidth = maNsc(Apo), binHeight = maNsr(Apo),
+     save.models = TRUE, robust = FALSE)
> ApoNr <- maNormNN(Apo[, 13], binWidth = maNsc(Apo), binHeight = maNsr(Apo),
+     save.models = TRUE, robust = TRUE)
> MM <- maM(Apo[maPrintTip(Apo) == pT, s])
> AA <- maA(Apo[maPrintTip(Apo) == pT, s])
> A <- seq(min(AA), max(AA), length = 100)
> lo <- loess(MM ~ AA, span = 0.4, degree = 1, family = "symmetric",
+     control = loess.control(trace.hat = "approximate", iterations = 5,
+         surface = "direct"))
> plot(AA, MM, pch = 20, xlab = "A", ylab = "M")
> lines(A, predictBias(A, ApoN$models, 1, pT), col = "green", lty = "longdash",
+     lwd = 2)
> lines(A, predictBias(A, ApoNr$models, 1, pT), col = "red", lwd = 2)
> lines(A, predict(lo, A), col = "blue", lwd = 2)
> legend(12.5, 1.5, legend = c("Loess", "NN", "Robust NN"), col = c("blue",
+     "green", "red"), lty = c("solid", "longdash", "solid"), cex = 1.1)
```

Note in Figure 6 that the robust neural network fitting is resistant to outlieres if compared with classical neural networks fitting. With respect to loess it has the advantage of providing a regression model easy to store.

References

M. J. Callow, S. Dudoit, E. L. Gong, T. P. Speed, and E. M. Rubin. Microarray expression profiling identifies genes with altered expression in hdl-deficient mice. *Genome Research*, 10:2022–2029, 2000.

S. Dudoit, J. P. Shaffer, and J. C. Boldrick. Multiple hypothesis testing in microarray experiments. *Statistical Science*, to appear, preprint available at UC Berkeley, Division Biostatistics working paper series: 2002-110, http://www.bepress.com/ucbbiostat/paper110, 2002.

A. L. Tarca, J. E. K. Cooke, and J. Mackay. Robust neural networks approach for spatial and intensity dependent normalization of cdna data. *Bioinformatics. submitted*, 2004.

P. H. Westfall and S. S. Young. *Resampling-based multiple testing: Examples and methods for p-value adjustment*. John Wiley & Sons, 1993.

Y. Yang, S. Dudoit, P. Luu, D. Lin, V. Peng, J. Ngai, and T. Speed. Normalization for cdna microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Res.*, 30:e15, 2002.

Y. Yang, S. Dudoit, P. Luu, and T. Speed. Normalization for cdna microarray data. *SPIE BiOS*, 2001.

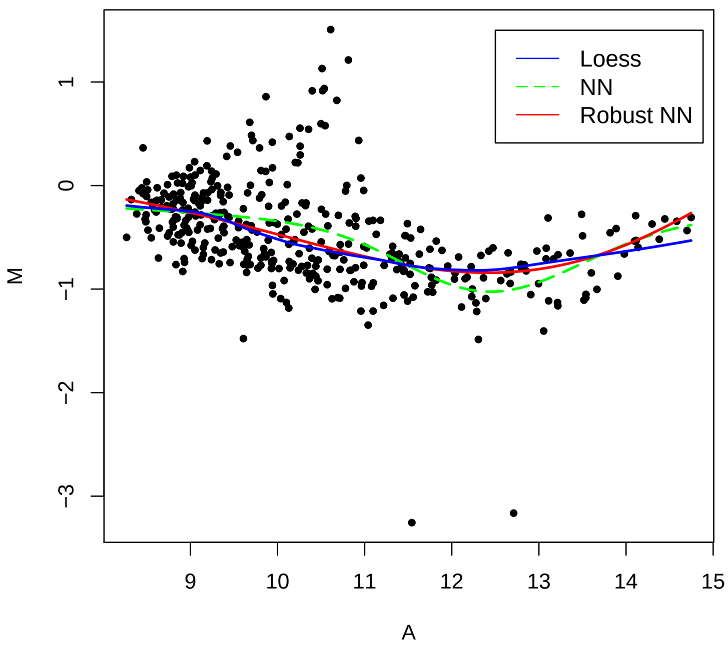

Figure 6: Bias estimates obtained with loess (continous blue), neural networks (dashed green), and robust neural network (continous red).