Bioconductor's nnNorm package

Adi Laurentiu Tarca[1]

March 28, 2007

Wayne State University, Medicine - Perinatology Research Branch, Detroit, Michigan, 48201.
`http://vortex.cs.wayne.edu/tarca`

1 Overview

The `nnNorm` package provides utilities to detect and correct for spatial bias with two color DNA microarrays (or paired single channel data). The normalization implemented in `nnNorm` package is based on neural networks models. Functionality to compare the distributions of the normalized log ratios is also provided. For the simpler case when only intensity normalization is desired (univariate distortion color model, similar to print tip loess normalization), we provide functionality to plot the bias estimate against the level of intensity for every print tip group.

Introduction to intensity and spatial normalization. This document provides a tutorial for using the `nnNorm` package. For a detailed description of the principles and algorithmic implemented by this package consult Tarca and Cooke (2005).

Spatial and intensity normalization implemented in `nnNorm`. The `nnNorm` package implements intensity and spatial normalization of cDNA data based on neural network models. In this approach, the log average intensity `A` and pseudo spatial coordinates `X` and `Y` (obtained by binning the print tip group space into small square regions), are used as predictors (inputs) into a neural network model. Resistance to outliers is enhanced by implementing a N-fold cross validation (default nFolds=10) in which the data for a current spot is not used to estimate its own bias but the one of its neighbours.

2 Changes with respect to previous versions

The first version of `nnNorm` was 1.4.0 and released within bioC 1.7. Thanks to the users' feed-back, `nnNorm` continued to improve through the subsequent versions. We would like to thank especially to: Ken Kompass (Wash Univ, St Louis, MO, USA) for pointing out the need of a verbose argument to the maNormNN function; Elizabeth Thomas (CHR, Harvard University) and Philippe Hupé (Institut Curie, Paris France) who provided us with data sets on which we could make improvements in the implementation of the algorithm. The changes in the actual `nnNorm` version with respect to the previous ones include: a) adding functionallity to detect spatial bias within the print tips, and b) modifying the normalization function `maNormNN` by: 1) using a N-fold cross-validation method to estimate the bias, where the user can choose the number of folds (current default is nFolds=10); 2) avoiding convergence problems in neural network fitting by running 5 different training sessions and taking the median over the 5 different estimates, 3) allowing for user specified weights to discard certain spots from the normalization 4) removing the "robust" argument which initially was designed to assign different weights to each spot, weights that were computed as a function of how extreme the log-ratio of the spot is for a given intensity range.

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`.

3 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 read existing data from an internet connection.

```
> 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 corrected red and green intensities.
Due to several factors, such as printing and washing, the raw log-ratios may depend not only on the average log-intensity but also on the spatial coordinates Yang et al. (2002).

The new function `detectSpatialBias` added in this version of `nnNorm` allows to identify automatically which print tips in the different arrays show spatial bias.

```
> tmp <- detectSpatialBias(Apo)

Processing array 1 of 16
****************
Processing array 2 of 16
****************
...
Processing array 16 of 16
****************

> tmp

$biasRow
           c1R c2R c3R c4R c5R c6R c7R c8R k1R k2R k3R k4R k5R k6R k7R k8R
```

```
PrintTip1    14   0   0   5   0   0   5   0  38   0   0  19   5   0   0   0
PrintTip2     5   0  10   0   5   5   0   0   5   0   0   0   0   0   0   0
PrintTip3     5   0  24   0   0   0   0   0   0   0   0   5   0   0   0   0
PrintTip4    19   5   5   0   0   5   0  14   5   0  10   5   0   0   0   0
PrintTip5    10   0   5   0  14  10   0   5  10   5   0   5   5   0   0   5
PrintTip6    19   5   5   0   0   0  10   0  10   0   0  10  10   0   5   0
PrintTip7    24   0  14  10   5  24   5  10  29   5  19  24  33   5  10   5
PrintTip8    10   5   5   0  14  19   0  14  24  14   5   5   0  10  10   5
PrintTip9    29   0   5   5   0   0   0   0  33  10  10  43   5   0   0   0
PrintTip10   24  14   5  24  10   5  14  10  19  10  14  19  10  19   5  10
PrintTip11   29  10  14   0   0  10  10  10  14  10  10  33  10   5   0   5
PrintTip12   24  29  10   5   0  33   0   5  14  10  19  29   0   5   0   0
PrintTip13   10  10  19  24  24  10  10  24   0  10  14   0  29  19  24  38
PrintTip14   19   0  24  43  29  29  29  52  33  14  14   0  24  14  33  38
PrintTip15   38   0  19  19  29  10  24  62  19  19   0  14   5  10  24  62
PrintTip16    5   0   0   0  10   0  24  14   0   0   5   0   0   5   5   5

$biasCol
            c1R c2R c3R c4R c5R c6R c7R c8R k1R k2R k3R k4R k5R k6R k7R k8R
PrintTip1     5   0   5   5   5  16   5  11  11   5   0  11  16   5  11   5
PrintTip2     5   5   5   0   5  16  11   5   5   5  11   5   0  11   5   5
PrintTip3     5   0  11   0   5   0   5   5   5   0   0   0  11  16  11  11
PrintTip4     0   0   0   0  16  16   5   5   5   0   5   5   5   5  11  11
PrintTip5    11  11  16   5  11  11   5  16  21   5  16  11  11   5   0  11
PrintTip6     0   5   5   0  11  11   5   5   0   5  16   5   5   0   5   5
PrintTip7     0   5   5  11   5  11  16   5   0  11  11  16   0  11   5  11
PrintTip8     5   5   0   5   5   5  16   5   5   5   5  11   0   5   5   0
PrintTip9     5   5   0   5  26   0   5   5  32   0   5   5  11   0   5   0
PrintTip10    5   0   5   0   0  16   5   5  11   5  26   5  11   0   5   5
PrintTip11    0   0   0   0   5   0   5   5   5   5   0   0   5   0   0   0
PrintTip12   58   5   5   0   5   5   5   5  11   5  11  11   5  11   0   5
PrintTip13    5   0   0  11  16   5   5   5   5   5   0   5   5  11  11   5
PrintTip14   21   0   5   5   5   5   5   5   5   5  11   5   0   0   5  11
PrintTip15    0   0  11   5   0  16   5   5  11   5   5  16   5   0   5  11
PrintTip16   16   5   5   5   5   5   5   5  16   5   0   5   5  11  16   5
```

The `detectSpatialBias` function computes two matrices: `biasRow` and `biasCol`. The elements of these matrices represent the fraction of rows (columns) for which the correlation coefficient between log-ratios, M, and column index (row index) is higher than a user specified treshold (default is 0.6). As it can be seen, the arrays 8 (c8R) and 16 (k8R) shows strong column-wise bias (log-ratios are correlated with the row index) for several print-tips (especially 13 to 15).

Similar information (but more time consuming) can be obtained by looking at the spatial distribution of M values in each array.

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))
```

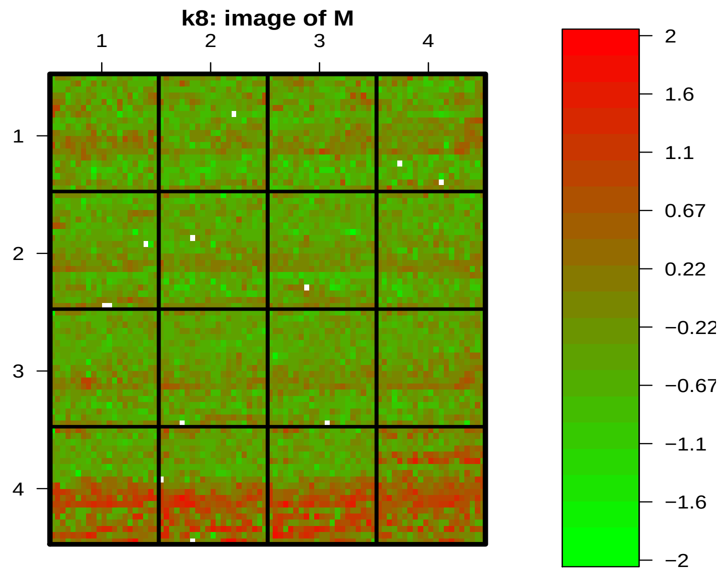

Figure 1: Image of raw M values for Apo data set, slide No. 16.

Observe the 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 `marrayNorm` type object (containing the normalized log ratios). The `maNormNN` function is a print-tip oriented normalization method.

```
> ApNN2DA <- maNormNN(Apo)
```

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 approaches that takes into account intensity and spatial bias fluctuations, like for e.g. a composite normalization method available in `marray` package.

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

We define a function able to compute the Westfall and Young (1993) adjusted p-values for different normalizations of the same Apo data. For this we use functionality of the `multtest` package (Dudoit et al. (2002)).

```
> library(multtest)
> getP <- function(maObj) {
```

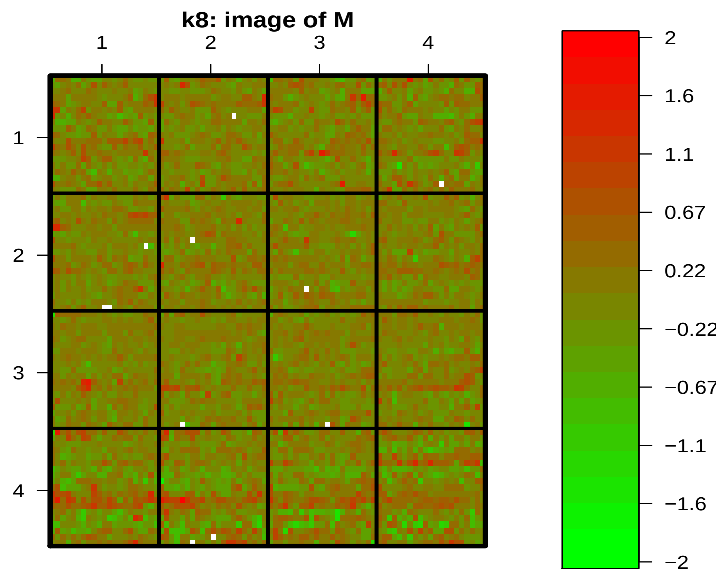

Figure 2: Image of M values after nnNorm normalization, for Apo data set, slide No. 16.

```
+     X <- maM(maObj)
+     class <- c(rep(0, 8), rep(1, 8))
+     resT <- mt.maxT(X, class, B = 15000)
+     ord <- order(resT$index)
+     resT$adjp[ord]
+ }
```

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

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

We plot 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)
> rnds <- sample(seq(-0.05, 0.05, length = 8))
> methods <- c("none", "cPLo2D", "pNN2DA")
> plot(rep(1, length(pAnone) - length(kout)), pAnone[-kout], xlim = c(1,
+     3), ylim = c(-4, 0), xlab = "Method", ylab = "log10(p)",
+     axes = FALSE)
> points(rep(1, 8) + rnds, pAnone[kout], col = "red", pch = 20)
> points(rep(2, length(pAcPLo2D) - length(kout)), pAcPLo2D[-kout])
> points(rep(2, 8) + rnds, pAcPLo2D[kout], col = "red", pch = 20)
```

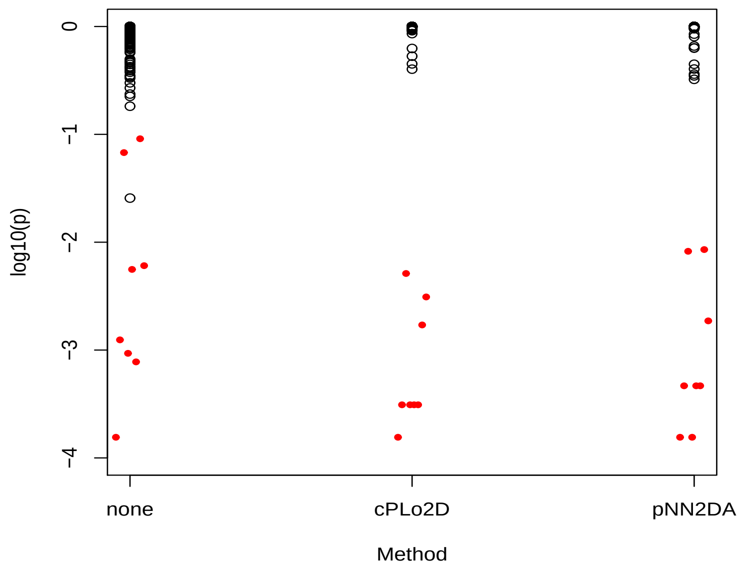

Figure 3: Adjusted p values for raw data (none), composite normalization (cpLo2D), and robust neural networks (pNN2DA)

```
> points(rep(3, length(pApNN2DA) - length(kout)), pApNN2DA[-kout])
> points(rep(3, 8) + rnds, 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) and are supossed to show down-regulation in this comparison.

Observe in Figure 3 that the 8 genes (represented by filled red circles) may be confidently distinguished from the bulk of non-differentially expressed genes at a threshold of about p=0.01. Now we may want to compare the ability of this new normalization method to reduce the variability of log ratios within slides. An easy way to look at the distributions of the normalized `M` values is to use the method `compNorm` of `nnNorm` package (see Figure 4).

```
> 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, as shown 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 to NULL, will discard the variable `Y` in the bias estimation model.

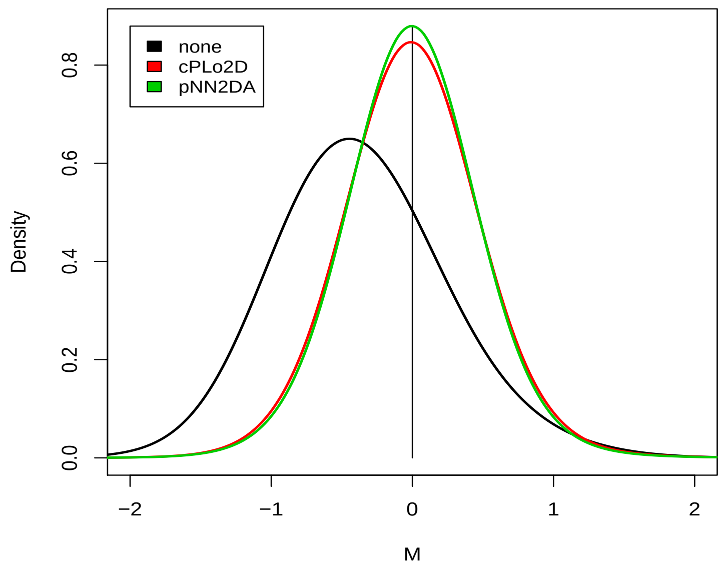

Figure 4: Density plots of normalized log ratios

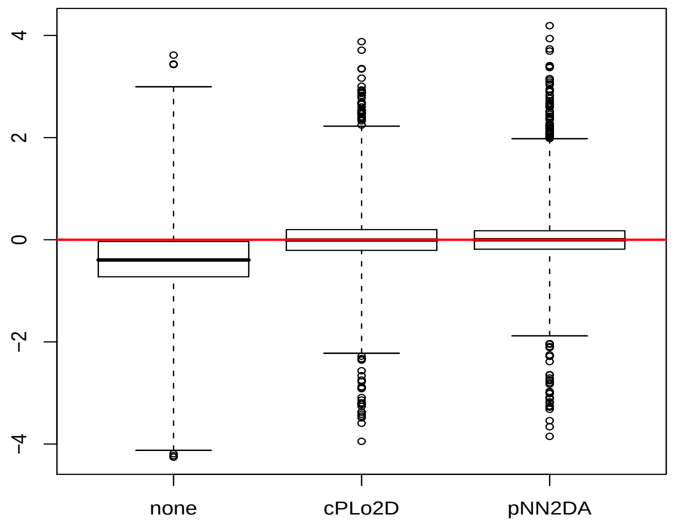

Figure 5: Box plot of normalized log ratios

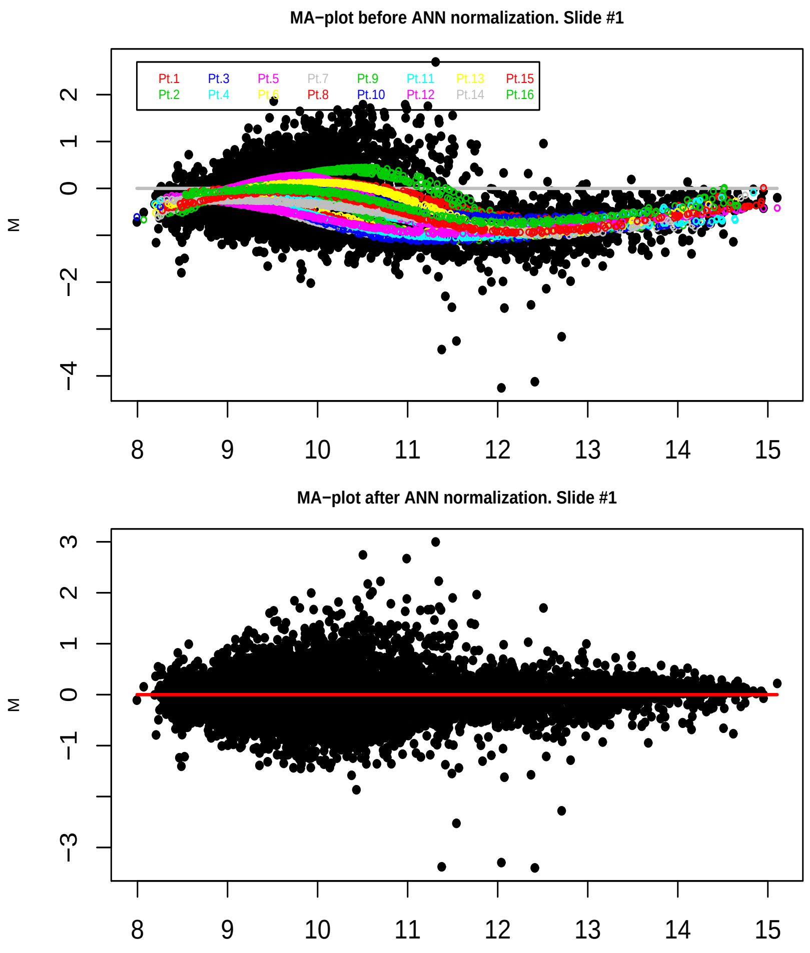

Figure 6: M-A plots before and after normalization for the whole slide.

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

In this case only `A` and `X` will be used as regressors for the bias. Similarly by setting `binWidth=NULL` in the same function will suppress the contribution of the `X` variable in 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 desired (`X`, and `Y` regressors discarded) we provide the facility to plot the bias estimates against the intensity level. For e.g. lets' perform intensity normalization only for the 13th slide of the Apo batch:

```
> par(mfrow = c(2, 1))
> par(mar = c(2, 4, 2, 2))
> ApoNr <- maNormNN(Apo[, 13], binWidth = NULL, binHeight = NULL,
+     maplots = TRUE)
```

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 and J. E. K. Cooke. A robust neural networks approach for spatial and intensity-dependent normalization of cdna microarray data. *Bioinformatics*, 21(11):2674–2683, 2005.

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, and T. Speed. Normalization for cdna microarray data. *SPIE BiOS*, 2001.

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.