Description of RPA (1.1.2)

Leo Lahti
Department of Information and Computer Science
TKK Helsinki University of Technology
leo.lahti@tkk.fi

October 28, 2009

1 Introduction

The *RPA* package[1] (Robust Probabilistic Averaging) provides tools for the analysis of probe reliability and differential gene expression on Affymetrix oligonucleotide arrays.

RPA estimates the reliability of individual probes on short oligonucletide arrays based on a probabilistic model for probe-level measurements described in Lahti et al.. Estimation probe reliability is directly based on probe-level observations, and independent of external information. RPA also gives a probeset-level estimate of differential gene expression between a user-specified control array and the other arrays in the data set, which makes the method applicable for preprocessing purposes for differential gene expression analysis.

A probe-level observation of differential gene expression for probe j is modeled as a sum of probeset-level differential gene expression vector $\mathbf{d}$ over the arrays, and probe-specific zero-mean Gaussian noise with variance τ_j^2. In practice, $\mathbf{d}$ and the probe-specific variances $\{\tau_j\}_{j=1}^{P}$ for the P probes within the probeset are estimated simultaneously using an iterative optimization scheme. The probe-specific inverse variance $1/\tau_j^2$ provides a measure of probe reliability.

Probabilistic formulation of the model allows incorporation of prior information concerning probe reliability into the analysis. With large sample sizes the solution converges to estimating the mean of the probe-level observations, weighted by probe reliability.

Emphasis on probe reliability analysis distinguishes RPA from probe-level preprocessing methods such as dChip's MBEI (Li and Wong, 2001), RMA (Irizarry et al., 2003), or FARMS Hochreiter et al. (2006). The RPA model and its relationship to other probe-level preprocessing methods is described in more detail in Lahti et al.. A key feature of RPA is that is utilizes probe-level measurements of differential expression to

[1] http://www.cis.hut.fi/projects/mi/software/RPA/

avoid the need to model unidentifiable probe affinities, the key probe-specific parameter in many preprocessing methods.

The *RPA* package utilizes the *affy* package Gautier et al. (2004) to handle probe-level data. For details about short oligonucleotide arrays and the design of the Affymetrix GeneChip arrays, we refer to the Appendix of the Affymetrix MAS manual Affymetrix (2001).

2 Using RPA to analyze probe reliability

In this example we use the `Dilution` dataset provided by *affydata* package. Note that this is a toy example with artificially small dataset for probe reliability analysis (4 arrays). A larger sample size is recommended for practical applications.

Start the analysis by loading the data:

```
> require(affy)
> require(affydata)
> data(Dilution)
```

RPA.pointestimate is the main function for performing RPA analysis. Let us perform the analysis for particular probesets in the Dilution data using the first array ($cind = 1$) as the control for calculating differential expression values for the other arrays.

```
> require(RPA)
> sets <- geneNames(Dilution)[1:5]
> rpa.results <- RPA.pointestimate(Dilution, sets, cind = 1)
```

Probe reliability and differential gene expression analysis can be performed for the whole data set with the following command. Note that this may take for a long time, depending on the size of the data set.

```
> rpa.results <- RPA.pointestimate(Dilution, cind = 1)
```

The results for one of the probe sets can be retrieved and visualized with

```
> set <- rownames(rpa.results$d)[[1]]
> d <- rpa.results[[set]]$d
> s2 <- rpa.results[[set]]$sigma2
> plot(rpa.results[set, ])
```

The output is shown in Figure 1.

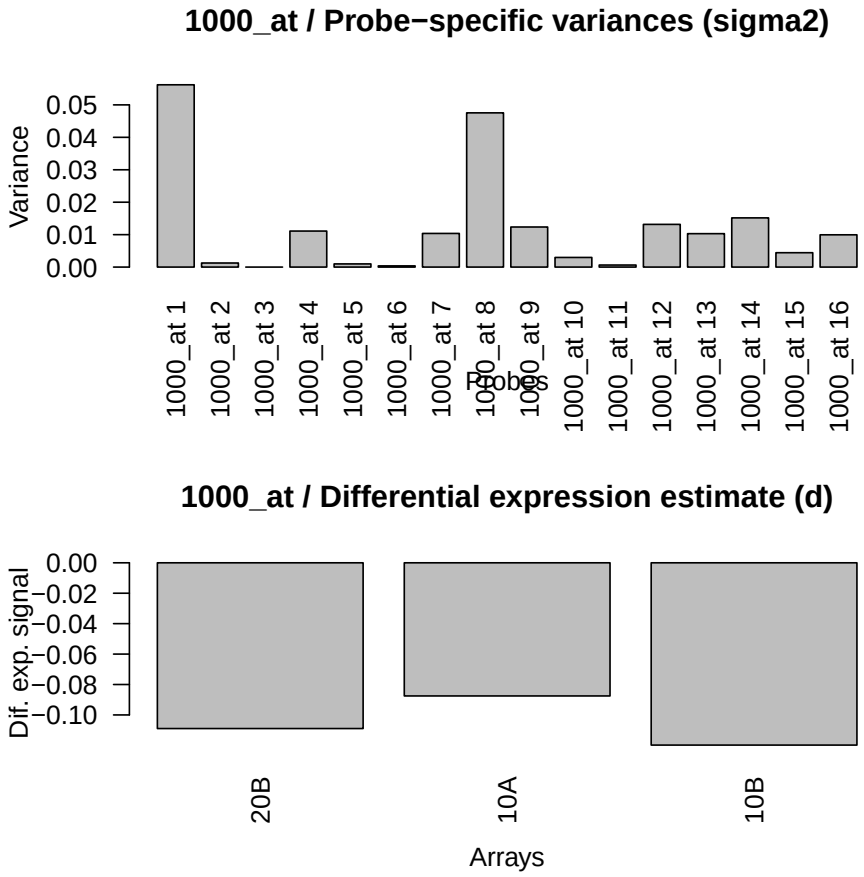

Figure 1: Estimates of probe-specific variances and differential gene expression for one of the probe sets.

2.1 Manual analysis of a single probe set

Individual probe sets can be analyzed with the function *RPA.iteration*. However, the data needs to be preprocessed before the analysis:

```
> Smat <- RPA.preprocess(Dilution, cind = 1)
```

Now we can pick probe-level data for the probe set '1000_at':

```
> set = "1000_at"
> S <- t(Smat$fcmat[pmindex(Dilution, set)[[1]], ])
```

Estimate probeset-level signal and probe-specific variances. Initialize by setting equal variance for the probes.

```
> res <- RPA.iteration(S, sigma2.guess = rep(1, ncol(S)))
```

2.2 Including probe-specific priors

The package allows user-specified priors for the probe-specific variances through the shape (alpha) and scale (beta) parameters for inverse Gamma distribution, which is the conjugate prior for the variances. By default, noninformative priors are assumed.

To include priors, start by creating a template for prior parameters. This sets default priors for the specified probesets:

```
> my.priors = initialize.priors(Dilution, sets, alpha = 1e-06,
+     beta = 1e-06)
```

Modify the prior template to set user-specified priors the probes. High values imply unreliable probes. Here we modify the priors for only one of the probes:

```
> set = "1000_at"
> probe.idx = 5
> my.priors[[set]]$beta[[probe.idx]] = 10
> my.priors[[set]]$alpha[[probe.idx]] = 10
```

Run RPA using the predefined priors. Note that priors are only used with sigma2.method = "basic"

```
> rpa.results <- RPA.pointestimate(Dilution, sets, priors = my.priors,
+     sigma2.method = "basic", d.method = "basic")
```

This toy example would show the probe reliability values for the probeset where the user-specified prior was set for one of the probes:

```
> barplot(rpa.results[[set]]$sigma2)
```

3 Differential gene expression analysis

RPA provides probeset-level estimates of differential gene expression between a user-specified control array and the other arrays in the data set. This makes RPA applicable for preprocessing purposes for differential gene expression analysis. The function 'rpa2eset' can be used to coerce an rpa object into an ExpressionSet object to allow downstream analysis of the results using standard R/BioC tools for gene expression data.

```
> eset = rpa2eset(rpa.results)
```

4 General usage

While the main functionality of the package is designed for gene expression studies, the general functions are potentially useful also in other applications. Let us demonstrate the usage with toy data. Consider a set of 300 observations (arrays; d), measured using 11 probes with varying performance (i.e. having different variances $sigma2$).

```
> set.seed(24)
> d.true <- rnorm(300, mean = 0, sd = 4)
> sigma2.true <- c(1, 1, 1, 1, 2, 2, 3, 3, 3, 4, 6)
> S <- NULL
> for (s2 in sigma2.true) {
+     S <- cbind(S, rnorm(length(d.true), mean = d.true, sd = sqrt(s2)))
+ }
```

Then fit the model on the toy data in S:

```
> set.seed(234)
> res <- RPA.iteration(S, sigma2.guess = rep(1, ncol(S)))
```

Visual comparison of the true and estimated parameters can be obtained with

```
> par(mfrow = c(2, 2))
> plot(res$d, d.true, main = "True vs. estimated d")
> plot(res$sigma2, sigma2.true, , main = "True vs. estimated sigma2")
```

Note that the method is meant to be used with large sample sizes; with small sample sizes it is prone to local optima. The solution can be regularized by setting user-specified priors.

References

Affymetrix. *Affymetrix Microarray Suite User Guide*. Affymetrix, Santa Clara, CA, version 5 edition, 2001.

Laurent Gautier, Leslie Cope, Benjamin M. Bolstad, and Rafael A. Irizarry. affy–analysis of Affymetrix GeneChip data at the probe level. *Bioinformatics*, 20(3):307–315, 2004.

Sepp Hochreiter, Djork-Arne Clevert, and Klaus Obermayer. A new summarization method for affymetrix probe level data. *Bioinformatics*, 22(8):943–949, 2006.

Rafael A. Irizarry, Benjamin M. Bolstad, Francois Collin, Leslie M. Cope, Bridget Hobbs, and Terence P. Speed. Summaries of Affymetrix GeneChip probe level data. *Nucl. Acids Res.*, 31(4):e15, 2003.

Leo Lahti, Laura L. Elo, Tero Aittokallio, and Samuel Kaski. Probabilistic analysis of probe reliability in differential gene expression studies with short oligonucleotide arrays. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*. To appear.

Cheng Li and Wing Hung Wong. Model-based analysis of oligonucleotide arrays: Expression index computation and outlier detection. *Proc. Natl. Acad. Sci.*, 98:31–36, 2001.