ARSyN User's Guide

Maria J. Nueda

5 July 2011

1. Departamento de Estadistica e I.O. Universidad de Alicante, Spain.
mj.nueda@ua.es
2. Centro de Investigacion Principe Felipe, Valencia, Spain.
aconesa@cipf.es

Contents

1 Introduction

ARSyN (ASCA Removal of Systematic Noise) is a R package for filtering single and multi-series time course microarray experiments.
ARSyN is based on the ASCA model developed by Smilde et al. (2005) to remove structural noise from microarray datasets. ASCA combines ANOVA and PCA (Principal Components Analysis) to analyze multifatorial omics datasets. So far, ASCA has been used for exploratory analysis (Jansen et al., 2005) and for the identification of responsive genes in transcriptomics (Nueda et al., 2007). ARSyN takes advantage of the data decomposition provided by the ASCA model to develop a novel statistical framework for the preprocessing of microarray data. In brief, ARSyN uses the PCAs of the ANOVA parameters and residuals in the ASCA model to identify and separate noise from signal in microarray data. After this decomposition, the data elements of interest are joined back together to reconstruct a filtered gene expression matrix which is free of structural biases. The filtered matrix has two main advantages:

1. Extracts the relevant gene expression variation related to the controlled variables in the experimental design. This data element is obtained from the main principal components (PC) of the ANOVA parameters.

2. Is free of structural noise that can be associated to batch effects or to other non-traceable sources of variation. This data element is identified in the main PC of the residuals of the ANOVA model.

2 Getting started

The **ARSyN** package can be downloaded from `http://www.ua.es/personal/mj.nueda`.

```
> source("sourceARSyN.R")
```

3 The program

The main function is **ARSyN.R** that needs the following parameters:

- *data*: data matrix with gene expression data, genes in rows, arrays in columns.
- *Covariates*: data matrix with experimental design information. It must contain as many columns as arrays and as many rows as experimental factors. Each cell contains the value of the array in the experimental factor.
- *Interaction*: logical to indicate whether interaction/s between factors should be analyzed (TRUE, default).
- *Join*: logical to indicate whether interaction/s must be analysed jointly with the second factor (TRUE, default) or not (FALSE).
- *Variability*: parameter for PC selection of the main effects models. This is the desired proportion of variability explained for the principal components of the main effects (time and experimental group). Variability=0.75 by default, (Nueda et al.).
- *beta*: parameter for PC selection of the residual model. Components selected will be those that explain more than beta plus the average component variability computed as the total data variability divided by the rank of the matrix associated to the factor, default beta=2, (Nueda et al.).

ARSyN.R is a function with the following steps:

1. Creates the design matrices needed for ASCA.
2. Calls **ARSyNmodel**, that is a function that first executes ASCA with the number of componentes by default to estimate the variability of each submodel, and then executes ASCA with the appropriate parameters.
3. Computes the filtered matrix.

4 Example: Obtaining the filtered data matrix.

To show the performance of ARSyN we use a small simulated dataset with only 1000 genes from which the first 50 genes are differentially expressed genes (gene1 to gene50). In this data there are 3 time-points, 3 experimental groups and 4 replicates for each condition, so a total of 36 data for each gene. Data is affected by a batch effect.
Load the data in your workspace:

```
> X <- read.table(file = "example_data.txt", header = TRUE, row.names = 1)
> cov <- read.table(file = "example_cov.txt", header = FALSE, row.names = 1)
> edesign <- read.table("edesign.txt")
```

To obtain the data filtered with the parameters by default:

```
> X.filtered <- ARSyN(data = X, Covariates = cov)
```

This matrix can be analysed by other statistical Time Course Microarray methodologies as maSigPro (Conesa et al., 2006), *timecourse* (Tai and Speed, 2006, 2009) and EDGE (Storey et al., 2005). Here we show the application of maSigPro:

```
> library(maSigPro)
```

Load the experimental design and make the maSigPro design matrix:

```
> edesign <- read.table("edesign.txt")
> dis <- make.design.matrix(edesign)
```

First we apply maSigPro to the original data:

```
> p.0 <- p.vector(X, dis)
```

```
[1] "fitting gene 100 out of 1000"
[1] "fitting gene 200 out of 1000"
[1] "fitting gene 300 out of 1000"
[1] "fitting gene 400 out of 1000"
[1] "fitting gene 500 out of 1000"
[1] "fitting gene 600 out of 1000"
[1] "fitting gene 700 out of 1000"
[1] "fitting gene 800 out of 1000"
[1] "fitting gene 900 out of 1000"
[1] "fitting gene 1000 out of 1000"
```

```
> T.0 <- T.fit(p.0)
> get.0 <- get.siggenes(T.0, vars = "all", rsq = 0.7)
> rownames(get.0$sig.genes$sig.profiles)
```

```
[1] "gene1"  "gene6"  "gene31" "gene39" "gene41" "gene42" "gene43"
```

We can observe that there are only 7 genes detected.
Next, we apply maSigPro to the filtered data:

```
> p.1 <- p.vector(X.filtered, dis)
```

```
[1] "fitting gene 100 out of 1000"
[1] "fitting gene 200 out of 1000"
[1] "fitting gene 300 out of 1000"
[1] "fitting gene 400 out of 1000"
[1] "fitting gene 500 out of 1000"
[1] "fitting gene 600 out of 1000"
```

```
[1] "fitting gene 700 out of 1000"
[1] "fitting gene 800 out of 1000"
[1] "fitting gene 900 out of 1000"
[1] "fitting gene 1000 out of 1000"
```

```
> T.1 <- T.fit(p.1)
```

```
[1] "fitting gene 100 out of 114"
```

```
> get.1 <- get.siggenes(T.1, vars = "all", rsq = 0.7)
> rownames(get.1$sig.genes$sig.profiles)
```

```
 [1] "gene1"   "gene2"   "gene3"   "gene4"   "gene5"   "gene6"   "gene7"
 [8] "gene8"   "gene9"   "gene10"  "gene11"  "gene12"  "gene13"  "gene14"
[15] "gene15"  "gene16"  "gene17"  "gene18"  "gene19"  "gene20"  "gene21"
[22] "gene22"  "gene23"  "gene24"  "gene25"  "gene26"  "gene27"  "gene28"
[29] "gene29"  "gene30"  "gene31"  "gene32"  "gene33"  "gene34"  "gene35"
[36] "gene36"  "gene37"  "gene38"  "gene39"  "gene40"  "gene41"  "gene42"
[43] "gene43"  "gene44"  "gene45"  "gene46"  "gene47"  "gene48"  "gene49"
[50] "gene50"  "gene264" "gene347" "gene808" "gene901"
```

Now we can see that the 50 differentially genes are detected and there are 4 false negatives. Figure 1 shows the distributions of the p-values of the first step of maSigPro. We can observe an increase of the number of genes with low p-values which is consistent with a general removal of noise from the data.

```
> hist(p.0$p.vector, ylim = c(0, 800), main = "Original data",
+     xlab = "First step maSigPro p-values")
```

```
> hist(p.1$p.vector, ylim = c(0, 800), main = "ARSyN filtered data",
+     xlab = "First step maSigPro p-values")
```

References

A. Conesa, M. Nueda, A. Ferrer, and M. Talón. maSigPro: a method to identify significantly differential expression profiles in time-course microarray experiments. *Bioinformatics*, 22(9):1096–1102, 2006. URL `http://bioinformatics.oupjournals.org/cgi/content/abstract/22/9/1096`.

J. Jansen, H. Hoefsloot, M. Timmerman, J. V. der Greef, J. Westerhuis, and A. Smilde. ASCA: analysis of multivariate data obtained from an experimental design. *Journal of Chemometrics*, 19:469–481, 2005.

M. Nueda, A. Conesa, and A. Ferrer. ARSyN: a method for the identification and removal of systematic noise in multifactorial time-course microarray experiments.

M. Nueda, A. Conesa, J. Westerhuis, H. Hoefsloot, A. Smilde, M. Talón, and A. Ferrer. Discovering gene expression patterns in Time Course Microarray Experiments by ANOVA-SCA. *Bioinformatics*, 23(14):1792–1800, 2007.

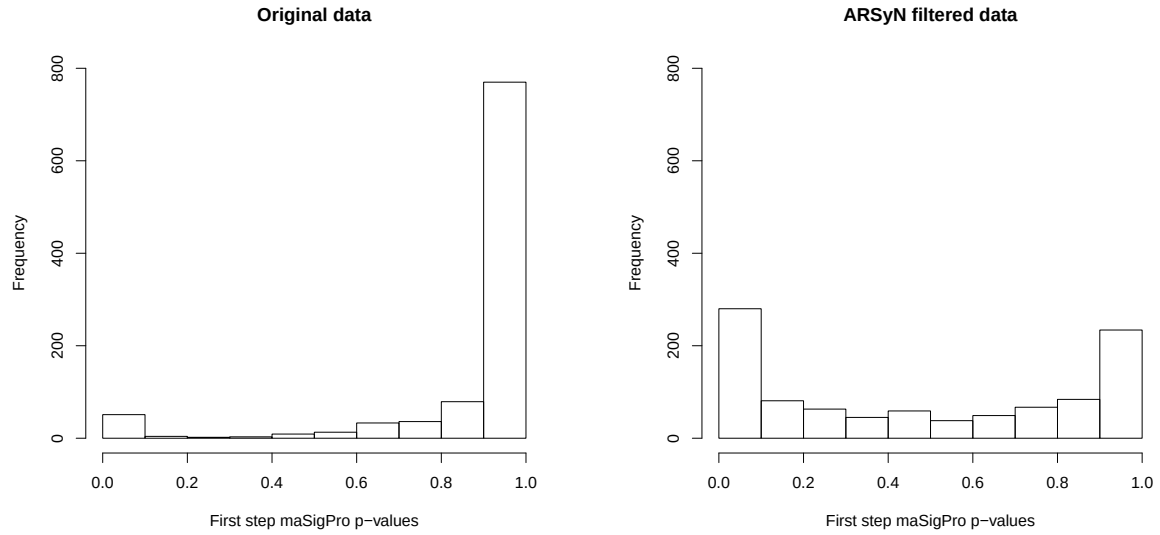

Figure 1: p-values distribution of the first step of maSigPro before and after applying the ARSyN filter.

A. Smilde, J. Jansen, H. Hoefsloot, R. Lamers, J. V. der Greef, and M. Timmerman. ANOVA-Simultaneous Component Analysis (ASCA): a new tool for analyzing designed metabolomics data. *Bioinformatics*, 21(13):3043–3048, 2005.

J. Storey, W. Xiao, J. Leek, R. Tompkins, and R. Davis. Significance analysis of time course microarray experiments. *PNAS*, 102(36):12837–12842, 2005.

Y. Tai and T. Speed. A multivariate empirical bayes statistic for replicated microarray time course data. *Annals of Statistics*, 34(5):2387–2412, 2006.

Y. Tai and T. Speed. On gene ranking using replicated microarray time course data. *Biometrics*, 65:40–51, 2009.