maSigPro User's Guide

Ana Conesa and Maria J. Nueda

30 July 2013

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

Contents

1 Introduction

maSigPro is a R package that initially was developed for the analysis of single and multiseries time course microarray experiments. Recently **maSigPro** has been adapted to deal also with Next Generation-Sequencing (NGS) series of data in a proper way. The usage of this opcion is explained in 5 section.
maSigPro follows a two steps regression strategy to find genes with significant temporal expression changes and significant differences between experimental groups. The method defines a general regression model for the data where the experimental groups are identified by dummy variables. The procedure first adjusts this global model by the least-squared technique

to identify differentially expressed genes and selects significant genes applying false discovery rate control procedures. Secondly, stepwise regression is applied as a variable selection strategy to study differences between experimental groups and to find statistically significant different profiles. The coefficients obtained in this second regression model will be useful to cluster together significant genes with similar expression patterns and to visualize the results. This document is a example-based guide for the use of **maSigPro**. We recommend to open a R sesion and go through this tutorial running the code given at the different sections. The guide does not provide a detailed description of the functions of the package or demonstrates the statistical basis of the methodology. The later is described in the work by (Conesa et al., 2006).

2 Getting started

The **maSigPro** package can be obtained from the Bioconductor repository or downloaded from `http://www.ua.es/personal/mj.nueda` and `http://bioinfo.cipf.es/downloads`. Load **maSigPro** by typing at the R prompt:

```
> library(maSigPro) # load maSigPro library
```

The on-line help of **maSigPro** can be started by typing at the R prompt:

```
>help(package="maSigPro") #for package help
>?p.vector #for function help
```

The analysis approach implemented in **maSigPro** is executed in 5 major steps (Figure 1) which are run by the package core functions **make.design.matrix()**, **p.vector()**, **T.fit()**, **get.siggenes()** and **see.genes()**. Additionally, the package provides the wrapping function **maSigPro()** which executes the entire analysis in one go.
In the following section we will explain the usage of each of these funcions using as example a data set from a multiple series time course experiment. At the end of this document we will also explain how to apply **maSigPro** to other experimental designs.

3 Multiple Series Time Course Experiment

For this section we will use a public data set from a plant abiotic stress study performed at the TIGR Institute by (Rensink et al., 2005). In this study, potato plants were subjected to three different types of abiotic stresses and gene expression was monitored at three time points after the start of the treatments. RNA was also collected from non-stressed plants at the same time points and all samples where hybridased against a common control on a 11K cDNA potato chip. There are three biological replicates for each experimental condition. For speed in this example we will use a random 1000 genes data subset of this study. This data set is part of the data supplied by the **maSigPro** package. The original data can be found at `http://www.tigr.org/tdb/potato/index.shtml`.

Before proceeding with this tutorial we will define some of the terms that will be used along this document. We denote experimental groups as the experimental factor for which temporal profiles are defined, like "*Treatment A*", "*Tissue1*", etc; conditions are each experimental group

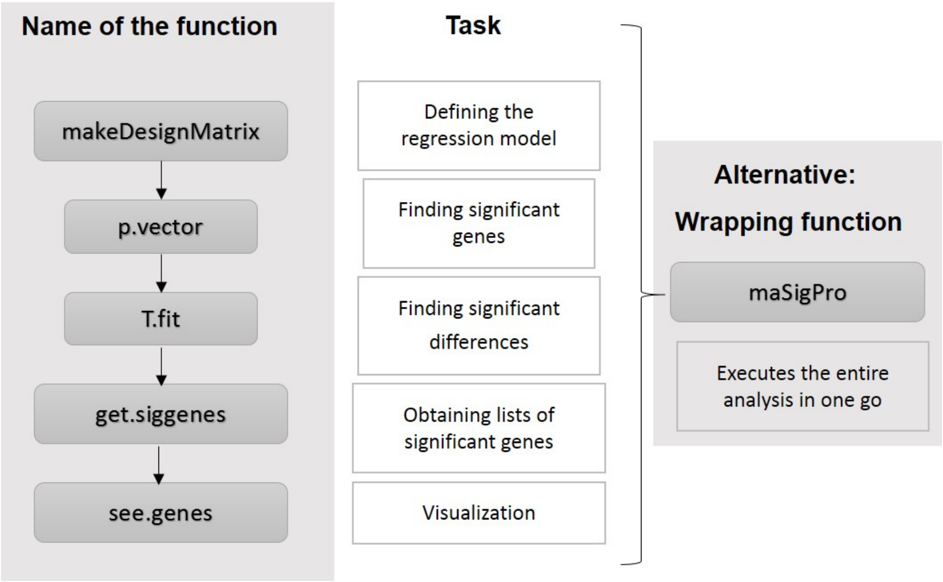

Name of the function
makeDesignMatrix
p.vector
T.fit
get.siggenes
see.genes
Task
Defining the regression model
Finding significant genes
Finding significant differences
Obtaining lists of significant genes
Visualization
Alternative:
Wrapping function
maSigPro
Executes the entire analysis in one go

Figure 1: maSigPro pipeline. Major functions.

vs. time combination like ”*Treatment A at Time 0*” conditions can have or not replicates. Variables are the regression variables defined by the **maSigPro** approach for the experiment regression model. **maSigPro** defines dummy variables to model differences between experimental groups. Dummy variables, Time and their interactions are the variables of the regression model. Load the data in your workspace:

```
> data(data.abiotic)
> data(edesign.abiotic)
```

The `edesign.abiotic` object describes the experimental design of this experiment in **maSigPro** format.

```
> edesign.abiotic
```

```
              Time Replicate Control Cold Heat Salt
Control_3H_1     3         1       1    0    0    0
Control_3H_2     3         1       1    0    0    0
Control_3H_3     3         1       1    0    0    0
Control_9H_1     9         2       1    0    0    0
Control_9H_2     9         2       1    0    0    0
Control_9H_3     9         2       1    0    0    0
Control_27H_1   27         3       1    0    0    0
Control_27H_2   27         3       1    0    0    0
Control_27H_3   27         3       1    0    0    0
Cold_3H_1        3         4       0    1    0    0
Cold_3H_2        3         4       0    1    0    0
Cold_3H_3        3         4       0    1    0    0
Cold_9H_1        9         5       0    1    0    0
```

```
Cold_9H_2     9      5     0   1   0   0
Cold_9H_3     9      5     0   1   0   0
Cold_27H_1   27      6     0   1   0   0
Cold_27H_2   27      6     0   1   0   0
Cold_27H_3   27      6     0   1   0   0
Heat_3H_1     3      7     0   0   1   0
Heat_3H_2     3      7     0   0   1   0
Heat_3H_3     3      7     0   0   1   0
Heat_9H_1     9      8     0   0   1   0
Heat_9H_2     9      8     0   0   1   0
Heat_9H_3     9      8     0   0   1   0
Heat_27H_1   27      9     0   0   1   0
Heat_27H_2   27      9     0   0   1   0
Heat_27H_3   27      9     0   0   1   0
Salt_3H_1     3     10     0   0   0   1
Salt_3H_2     3     10     0   0   0   1
Salt_3H_3     3     10     0   0   0   1
Salt_9H_1     9     11     0   0   0   1
Salt_9H_2     9     11     0   0   0   1
Salt_9H_3     9     11     0   0   0   1
Salt_27H_1   27     12     0   0   0   1
Salt_27H_2   27     12     0   0   0   1
Salt_27H_3   27     12     0   0   0   1
```

Note that arrays are given in rows and experiment descriptors are provided in columns. The first column shows the value that variable Time takes in each array. Replicates column is an index column that indicates the replicated arrays: all arrays belonging to the same experimental condition must be given the same number. The remaining columns are binary columns that give the assignment of arrays to experimental groups. There are as many binary columns as experimental groups and arrays take the value 1 or 0 whether they belong or not to that experimental group.

The `data.abiotic` object is a matrix with normalized gene expression data. Genes must be in rows and arrays in columns. **maSigPro** uses row and colunm names of the data and edesign objects throughout the package. Array names are the labels of the rows of the edesign object and columns of the data object that must be in the same order. GeneIDs are the labels of the rows in the data object. And experiment descriptors are put in the column names of the edesign object.

```
> colnames(data.abiotic)
> rownames(edesign.abiotic)
> colnames(edesign.abiotic)
> rownames(data.abiotic)
```

3.1 Defining the regression model

Create a regression matrix for the full regression model:

```
> design <- make.design.matrix(edesign.abiotic, degree = 2)
```

This example has three time points, so we can consider up to a quadratic regression model (`degree = 2`). Larger number of time points would potentially allow a higher polynomial degree. `design` is a list. Its element `dis` is the actual regression design matrix. `groups.vector` contains the assignment of regression variables to experimental groups.

```
> design$groups.vector

 [1] "ColdvsControl" "HeatvsControl" "SaltvsControl"
 [4] "Control"       "ColdvsControl" "HeatvsControl"
 [7] "SaltvsControl" "Control"       "ColdvsControl"
[10] "HeatvsControl" "SaltvsControl"
```

3.2 Finding significant genes

The next step is to compute a regression fit for each gene. This is done by the function **p.vector()**. This function also computes the p-value associated to the F-Statistic of the model, which is used to select significant genes. By default **maSigPro** corrects this p-value for multiple comparisons by applying the linear step-up (B-H) false discovery rate (FDR) procedure (Benjamini and Hochberg, 1995). This procedure can be modified by choosing another option of the **p.adjust** function that is controlled by the function parameter `MT.adjust` of **p.vector**. The level of FDR control is given by the function parameter `Q`.

```
> fit <- p.vector(data.abiotic, design, Q = 0.05, MT.adjust = "BH", min.obs = 20)
```

p.vector() returns a list of values:

```
> fit$i # returns the number of significant genes
> fit$alfa # gives p-value at the Q false discovery control level
> fit$SELEC # is a matrix with the significant genes and their expression values
```

3.3 Finding significant differences

Once significant genes have been found, **maSigPro** applies a variable selection procedure to find significant variables for each gene. This will ultimatelly be used to find which are the profile differences between experimental groups. This step is done by the **T.fit()** function.

```
> tstep <- T.fit(fit, step.method = "backward", alfa = 0.05)
```

T.fit() executes stepwise regression. The step.method can be ”backward” or ”forward” indicating whether the step procedure starts from the model with all or none variables. Use method ”two.ways.backward” and ”two.ways.forward” to allow variables to both get in and out. At each regression step the p-value of each variable is computed and variables get in/out the model when this p-value is lower or higher than the given cut-off value alfa. `tstep` is also a list. Its element `sol` is a matrix of statistical results obtained by the stepwise regression. For each selected gene the following values are given:

`p-value` of the regression ANOVA

`R-squared` of the model

`p-value` of the regression coefficients of the selected variables

3.4 Obtaining lists of significant genes

The following step is to generate lists of significant genes according to the way we want to see results. This is done by the function **get.siggenes()**. This function has two major arguments, `rsq` and `vars`.

`rsq`: is a cutt-off value for the R-squared of the regression model.

`vars`: is used to indicate how to group variables to show results. There are 3 possible values:

”`groups`”: This will generate a list of significant genes for each experimental group. The list corresponding to the reference group will contain genes whose expression profile is significantly different from a 0 profile. The lists corresponding to the remaining experimental groups will contain genes whose profiles are different from the reference group.

”`all`”: One unique list of significant genes a any model variable will be produced.

”`each`”: There will be as many lists as variables in the regression model. This can be used to analyze specific differences, for example genes that have linear or saturation kinetics.

```
> sigs <- get.siggenes(tstep, rsq = 0.6, vars = "groups")
> sigs$summary
```

The element `summary` is a data frame containing the significant genes for the selected `vars` You can further explore your results by:

```
> names(sigs$sig.genes)
```

```
[1] "Control"       "ColdvsControl" "HeatvsControl" "SaltvsControl"
```

```
> names(sigs$sig.genes$ColdvsControl)
```

```
[1] "sig.profiles"  "coefficients"  "group.coeffs"  "sig.pvalues"
[5] "g"             "edesign"       "groups.vector"
```

3.5 Graphical display

maSigPro has a number of functions for the visual exploration of the results.
suma2Venn() displays the `summary` result as a Venn diagram. For example, the following command will make a Venn diagram of the significant genes for the three stress experimental groups (Figure 2).

```
> suma2Venn(sigs$summary[, c(2:4)])
```

PlotGroups() creates a plot of gene expression profiles by experimental group. For example, STMDE66 is a gene which shows significant profile differences between the control and the cold and salt strees experimental groups, but not significant differences between control and heat experimental groups (Figure 3).

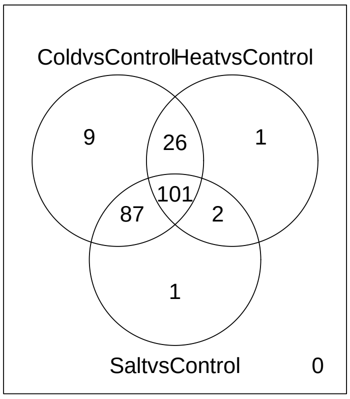


Figure 2: Venn Diagram for Cold, Heat and Salt stress significant genes

```
> STMDE66 <- data.abiotic[rownames(data.abiotic)=="STMDE66", ]
```

```
> PlotGroups (STMDE66, edesign = edesign.abiotic)
```

We can also add to the plot the regression curve computed for this gene.

```
> PlotGroups (STMDE66, edesign = edesign.abiotic, show.fit = T,
+ dis = design$dis, groups.vector = design$groups.vector)
```

Use **see.genes()** to visualize the result of a group of genes, for example, to visualize the significant genes of the *ControlvsSalt* comparison. **see.genes()** performs a cluster analysis to group genes by similar profiles. The resulting clusters are then plotted in two fashions: as experiment-wide expression profiles and as by-groups profiles. The first plot (Figure 4) will help to evaluate the consistency of the clusters while the second plot shows clearly the differences between groups (Figure 5).

```
see.genes(sigs$sig.genes$ColdvsControl, main = "ColdvsControl", show.fit = T,
          dis =design$dis, cluster.method="kmeans" ,cluster.data = 1, k = 9)
```

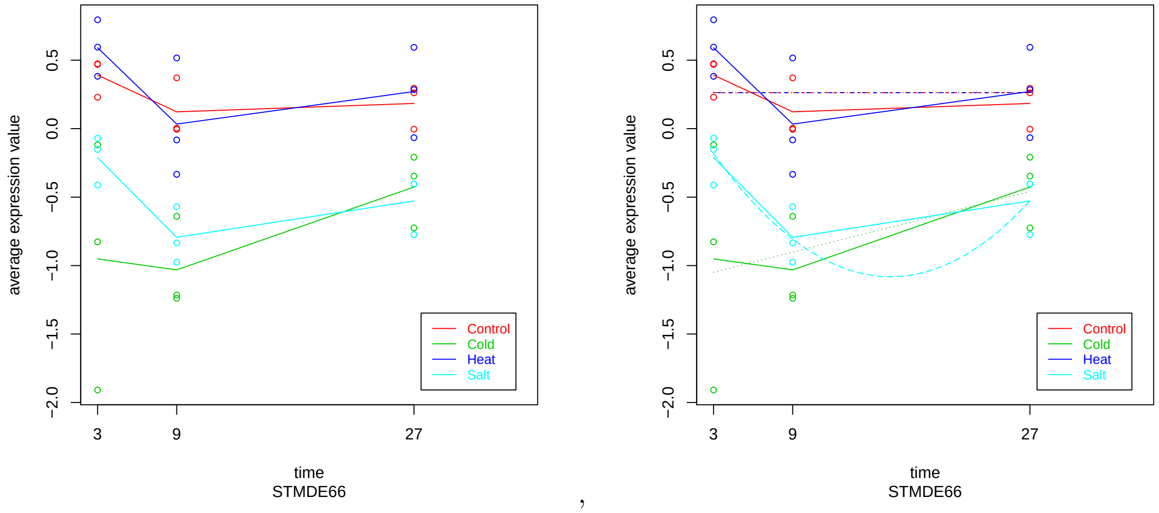
3.6 The maSigPro wrapper function

The five analyse steps decribed in the previous sections can be executed jointly by the function **maSigPro()**. The minimal sintax is:

```
> ts.analysis <- maSigPro (data.abiotic, edesign.abiotic)
```

Note that the default option directs the graphical output to a pdf. Set graphics off by `pdf = FALSE`. Other arguments are those of the different called functions. For example:

```
> ts.analysis <- maSigPro (data.abiotic, edesign.abiotic,
                           step.method = "two.ways.backward",
                           vars = "all", cluster.method = "hclust")
```



(a) , (b)

Figure 3: PlotGroups of gene STMDE66 without and with display or regression curves

4 Other designs

4.1 Single Series Time Course

The use of **maSigPro** in Single Series Time Course experiment is straightforward. Make a `edesign` object with just one group column containing all 1s and proceed as described above. Note that when using the **get.siggenes()** funcion the options ”all” and ”groups” of the argument `vars` will return the same result. You can use option ”each” to analyze the type of responses present in the significant genes: significant genes at the ”intercept” term will have a significant expression value at the starting time; genes associated to the variable ”Time” will have a significant linear component, which can be induction or repression depending on the sign of their coefficient; genes associated to the variable ”Time2” will show a change in the linear response that might be indicating transitory or saturation reponses, etc...
Here follows an example of a Single Series analysis.

```
## make a single series edesign
> Time <- rep(c(1,5,10,24), each = 3)
> Replicates <- rep(c(1:4), each = 3)
> Group <- rep(1,12)
> ss.edesign <- cbind(Time,Replicates,Group)
> rownames(ss.edesign) <- paste("Array", c(1:12), sep = "")
## Create data set
> ss.GENE <- function(n, r, var11 = 0.01, var12 = 0.02, var13 = 0.02,
           var14 = 0.02, a1 = 0, a2 = 0, a3 = 0, a4 = 0) {
  tc.dat <- NULL
  for (i in 1:n) {
```

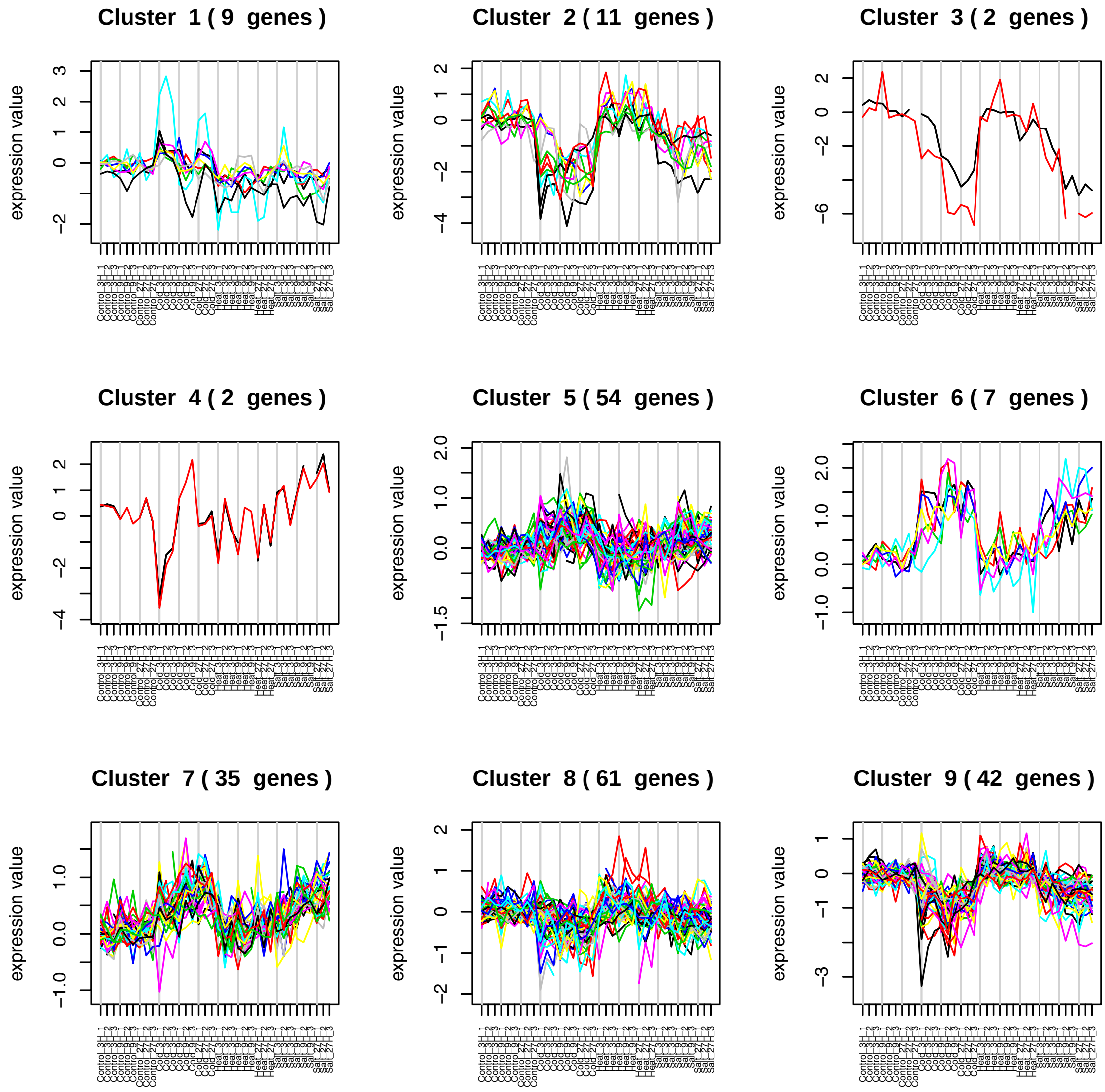


Figure 4: Cluster Analysis ColdvsControl significant genes

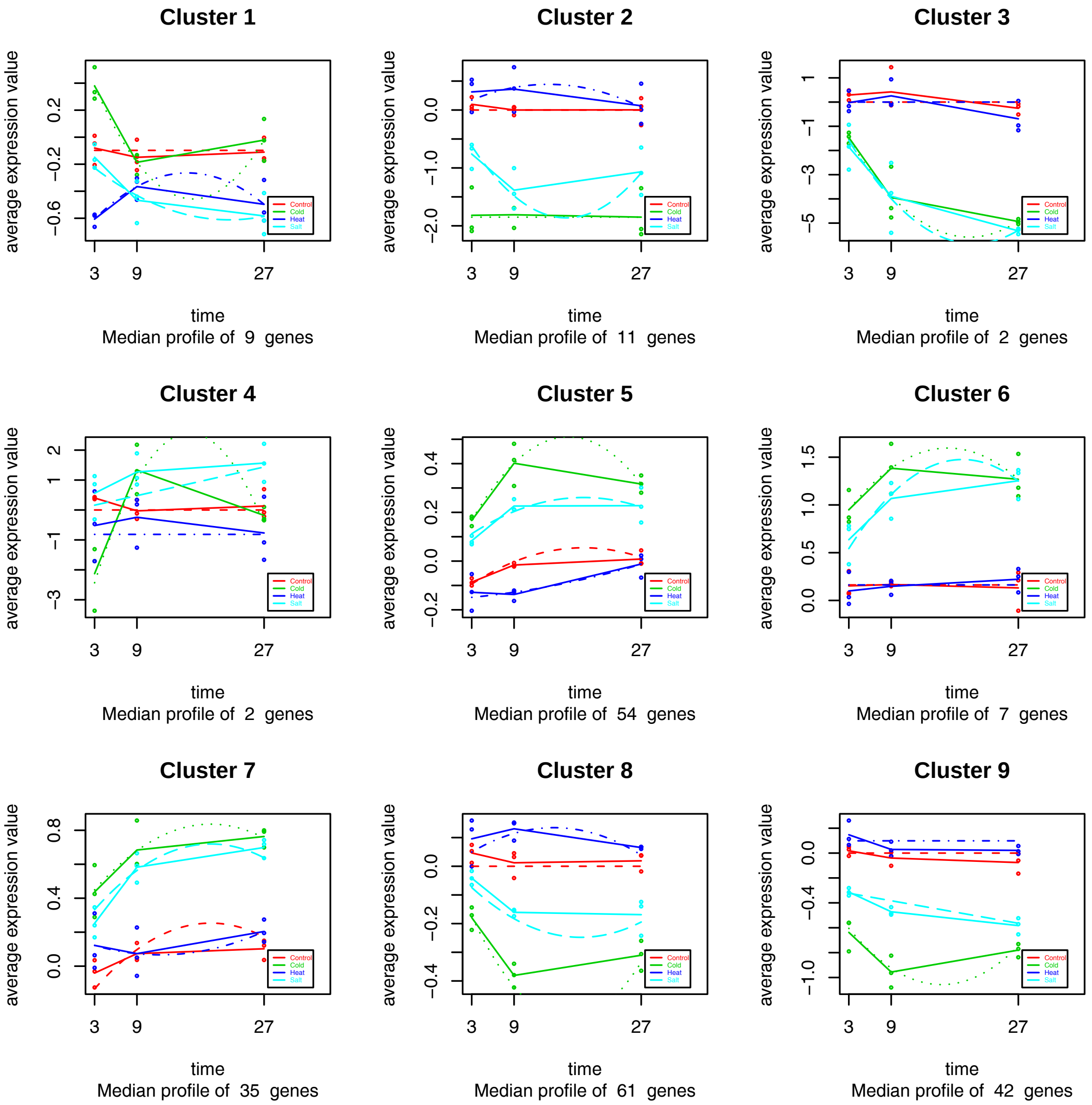


Figure 5: Expression Profiles ColdvsControl significant genes

```
    gene <- c(rnorm(r, a1, var11), rnorm(r, a1, var12),
              rnorm(r, a3, var13), rnorm(r, a4, var14))
    tc.dat <- rbind(tc.dat, gene)
  }
  tc.dat }
> flat <-ss.GENE(n = 85, r = 3) # flat
> induc <- ss.GENE(n = 5, r = 3, a1 = 0, a2 = 0.2, a3 = 0.6, a4 = 1) # induction
> sat <- ss.GENE(n = 5, r = 3, a1 = 0, a2 = 1, a3 = 1.1, a4 = 1.2) # saturation
> ord <- ss.GENE(n = 5, r = 3, a1 = -0.8, a2 = -1, a3 = -1.3, a4 =-0.9)  # intercept
> ss.DATA <- rbind(flat, induc,sat,ord)
> rownames(ss.DATA) <- paste("feature", c(1:100), sep = "")
> colnames(ss.DATA) <- paste("Array", c(1:12), sep = "")
# run maSigPro
> ss.example <- maSigPro(ss.DATA, ss.edesign, vars="each")
```

4.2 Common Starting Time

The following example illustrates how to build the edesign matrix when a common 0 time is applicable to the different experimental groups.

```
> data(edesignCT)
```

In this example Array1 and Array2 do not belong to any treatment. They are a common reference for all groups, values without any treatment at time 0.

4.3 Using Measured Parameters as Independent Variable

The regression approach followed by **maSigPro** allows that gene expression profile differences can not only be evaluated as a function of Time but another variables, such as a physiological parameter measured at the experimental condition, can be used as well as the independent variable.
The following example illustrates how to build the edesign matrix in such case. In this experiment gene expression was analyzed as a response to a temperature shift in a growing *E.coli* culture. The culture OD was measured for each sample.

```
> data(edesign.OD)
```

5 Next Generation-Sequencing series

maSigPro uses **lm()** function to fit a linear model where statistics for inference use normal distribution. This is the right treatment when dealing with normal distributed data or big samples. However, for non-normal small samples, results can not be correct.
The statistical distribution for tag counts data as RNA-Seq data may be Poisson or Binomial. However, the overdispersion of the data suggest the Negative Binomial (NB) can model these distributions in a better way. This distribution depends on a θ parameter to model the overdispersion that is related to the mean(μ) in the following way:

$$Y \sim NB(\theta), Var(Y) = \mu + \frac{\mu}{\theta}$$

maSigPro has been adapted to take into account non-normal distribution of the data. New arguments have been added in `p.vector` function to deal with this type of data in a proper way:

`counts`: a logical indicating whether your data are counts. By default is FALSE for microarray treatment.

`theta`: θ parameter for negative.binomial family. By default $\theta = 10$.

`family`: the distribution function to be used in the GLM. It must be specified as a function: gaussian(), poisson(), negative.binomial(theta)... If NULL family will be negative.binomial(theta) when counts=TRUE or gaussian() when counts=FALSE.

The recommended analysis to deal with RNA-Seq data is the GLM with negative.binomial family. θ must be specified and it can be computed by using available methods as edgeR ((Robinson et al., 2010)). The application of maSigPro with several values of θ to the same datasets did not reveal significant differences in gene selection. Taking this into consideration we have put by default $\theta = 10$ for being an average value. Moreover we give to the user the option of applying whatever exponential family to explore other possibilities.
`NBdata` is a subset of a bigger normalized dataset with 2 experimental groups, 6 time-points and 3 replicates. Simulation has been done by using a negative binomial distribution with $\theta = 5$ to illustrate this section. The first 20 genes are simulated with changes among time. `NBdesign` is the design matrix.

```
> data(NBdata)
> data(NBdesign)
```

```
> d <- make.design.matrix(NBdesign)
```

If we can use maSigPro with $theta = 10$:

```
> library(MASS)
> NBp <- p.vector(NBdata, d, counts=TRUE)
> NBt <- T.fit(NBp)
> get<-get.siggenes(NBt, vars="all")
> get$summary
```

These genes can be grouped in 4 clusters 6:

```
see.genes(get$sig.genes, k = 4)
```

If we can use maSigPro with a specific *theta*, in this case, $theta = 5$:

```
> NBp <- p.vector(NBdata, d, counts=TRUE, theta=5)
```

Also, a specific family, for instance, poisson can be specified. In such case counts and theta parameters are omitted.

```
> NBp <- p.vector(NBdata, d, family=poisson() )
```

Changing this arguments in p.vector is enough. Further functions will take these arguments from a `p.vector` object.

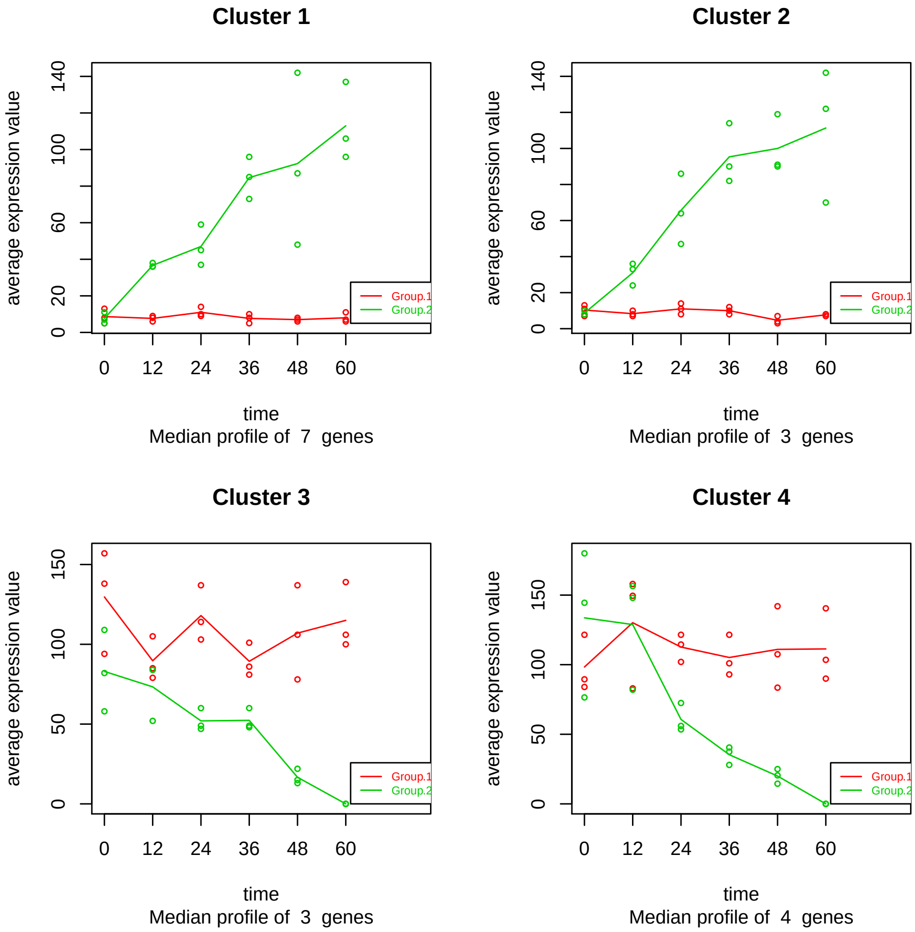


Figure 6: Cluster Analysis significant genes

References

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series B*, 57:289–300, 1995. URL `http://wwwtigr.org/tdb/potato/index.shtml`.

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`.

W. Rensink, S. Iobst, A. Hart, S. Stegalkina, J. Liu, and C. Buell. Gene expression profiling of potato responses to cold, heat and salt stress. *Funct Integr Genomics*, 5(4):201–207, 2005. URL `http://wwwtigr.org/tdb/potato/index.shtml`.

M. Robinson, D. McCarthy, and G. Smyth. edgeR: a Bioconductor package for differential expres-sion analysis. *Bioinformatics*, 26:139–140, 2010.