maSigPro User's Guide

Ana Conesa[1] and Maria José Nueda[2]

9 September 2006

1. Centro de Genómica. Valencia Institute for Agricultural Research,Moncada,Spain.
aconesa@ivia.es
2. Department of Statistics. University of Alicante, Alicante,Spain.
mj.nueda@ua.es

Contents

1 Introduction

`maSigPro` is a R package for the analysis of single and multiseries time course microarray experiments. **maSigPro** follows a two steps regression strategy to find genes with significant temporal expression changes and significant differences between experimetal 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 turorial running the code given at the different sections. The guide does not provides 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. (2005). The on-line help of `maSigPro` can be started by typing at the R promt:

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

maSigPro is a regression based approach. Linear regression methods for the analysis of microarray data can be found in the package `limma`. `limma` combines ANOVA with an empirical Bayes stragegy for the assessment of differencial expression in designed microarray experiments. The `limma` User's Guide provides some examples on how use this approach for the analysis of time series by multiple pair-wise comparisons. `maSigPro` uses a different approach to model expression profiles and to assess significant profile differences between experimental groups. `GeneTS` is also a package for analysis of time course data to be used when the aim of the experiment is to infere gene interaction networks.

2 Getting started

The `maSigPro` package can be obtained from the Bioconductor repository or downloaded from `http://www.ivia.es/centrogenomica/bioinformatics.htm`. Load `maSigPro` by typing at the R prompt:

```
> library(maSigPro)
```

The `maSigPro` vignette will be added to the Vignettes menu of R.

```
> shell.exec("./library.maSigPro/doc/maSigPro-tutorial.pdf")
```

will open this document.

The analysis approach implemented in `maSigPro` is executed in 5 major steps which are run by the package core functions **make.design.matrix()**, **p.vector()**, **T.fit()**, **get.siggenes()** and **see.genes()**. Additionaly, 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 exprimental 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* 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 work space:

```
> 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` 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 and columns of the `edesign` and `data` objects respectively, and 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 `edesign`.

```
> 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 matrix. `groups.vector` contains the assignment of regression variables to experimental groups.

```
> design$dis
```

```
> design$groups.vector
```

```
 [1] "ColdvsControl" "HeatvsControl" "SaltvsControl" "Control"
 [5] "ColdvsControl" "HeatvsControl" "SaltvsControl" "Control"
 [9] "ColdvsControl" "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. `maSigPro` corrects this p-value for multiple comparisons by applying false discovery rate (FDR) procedures. The level of FDR control is given by the function paramter `Q`.

```
> fit <- p.vector(data.abiotic, design, Q = 0.05, 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 containing 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 Obtaning 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 (1).

```
> 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 (5).

```
> 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)
```

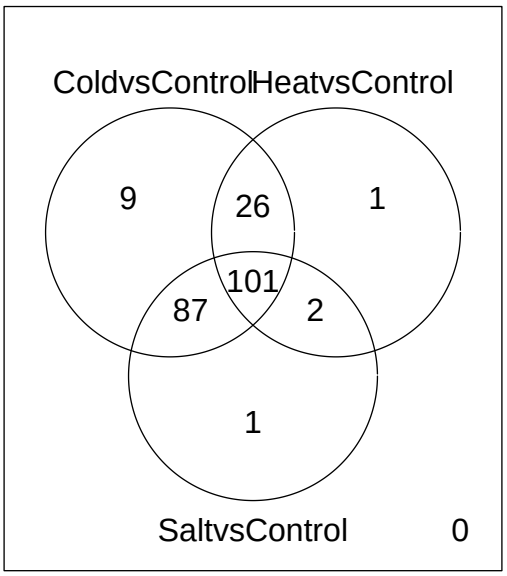
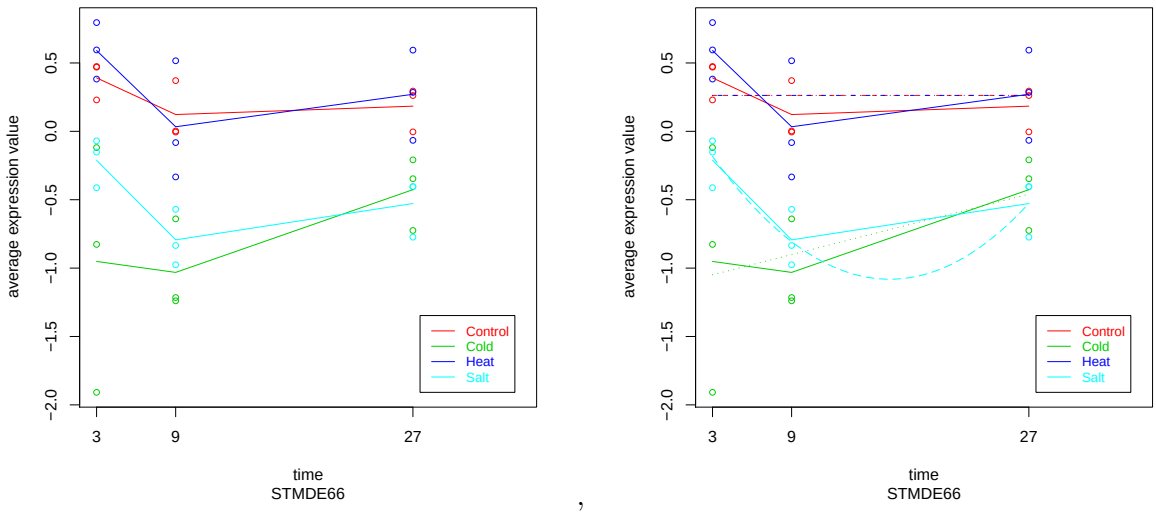


Figure 1: Venn Diagram for Cold, Heat and Salt stress significant genes



(a) , (b)

Figure 2: PlotGroups of gene STMDE66 without and with display or regression curves

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()** creates firstly a heatmap of the selected genes and the performs a cluster analysis to group genes by similar profiles (3). The resulting clusters are then plotted in two fashions: as experiment-wide expression profiles and as by-groups profiles. The first plot (4) will help to evaluate the consistency of the clusters while the second plot shows clearly the differences between groups 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")
```

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`; genen 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) {
```

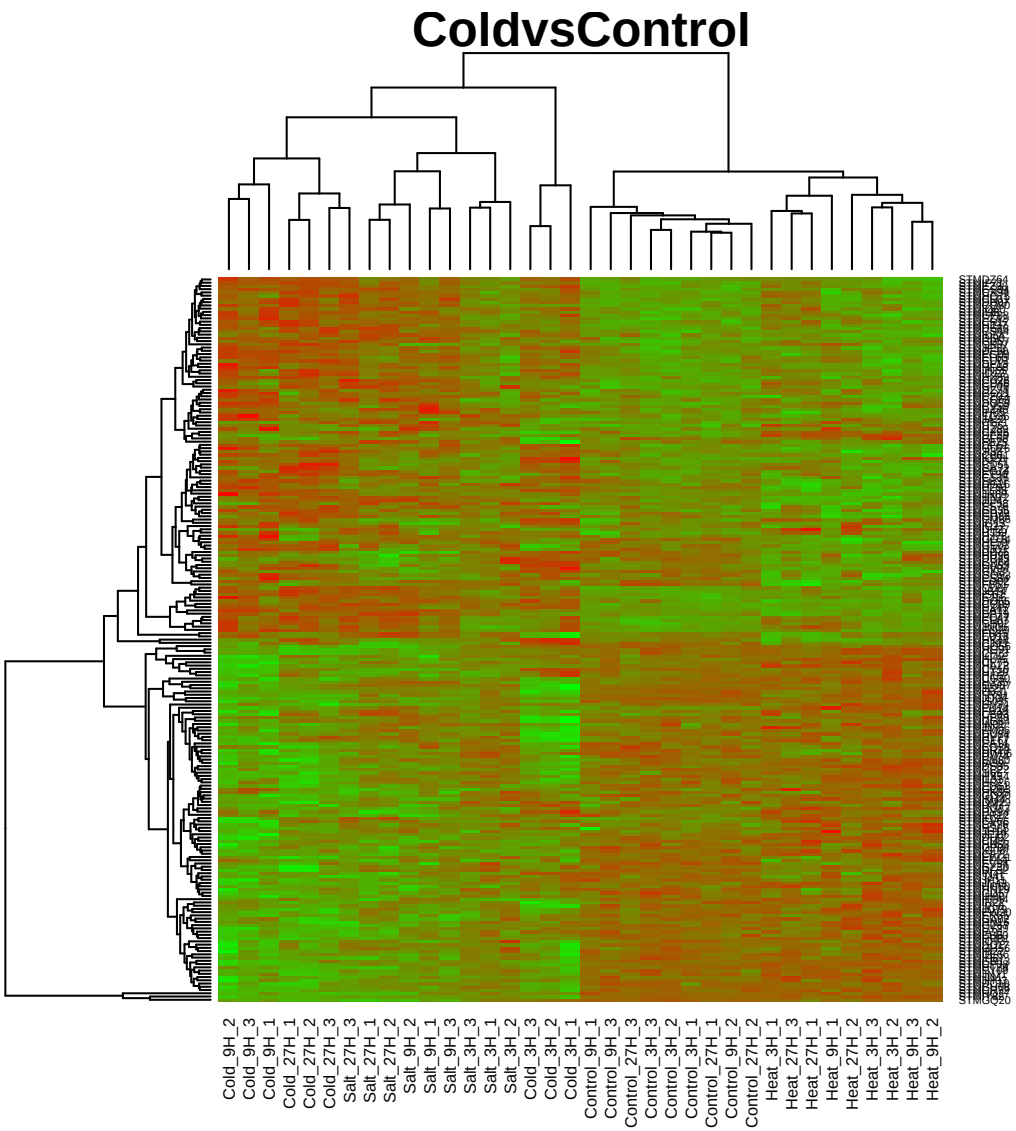


Figure 3: Heatmap ColdvsControl significant genes

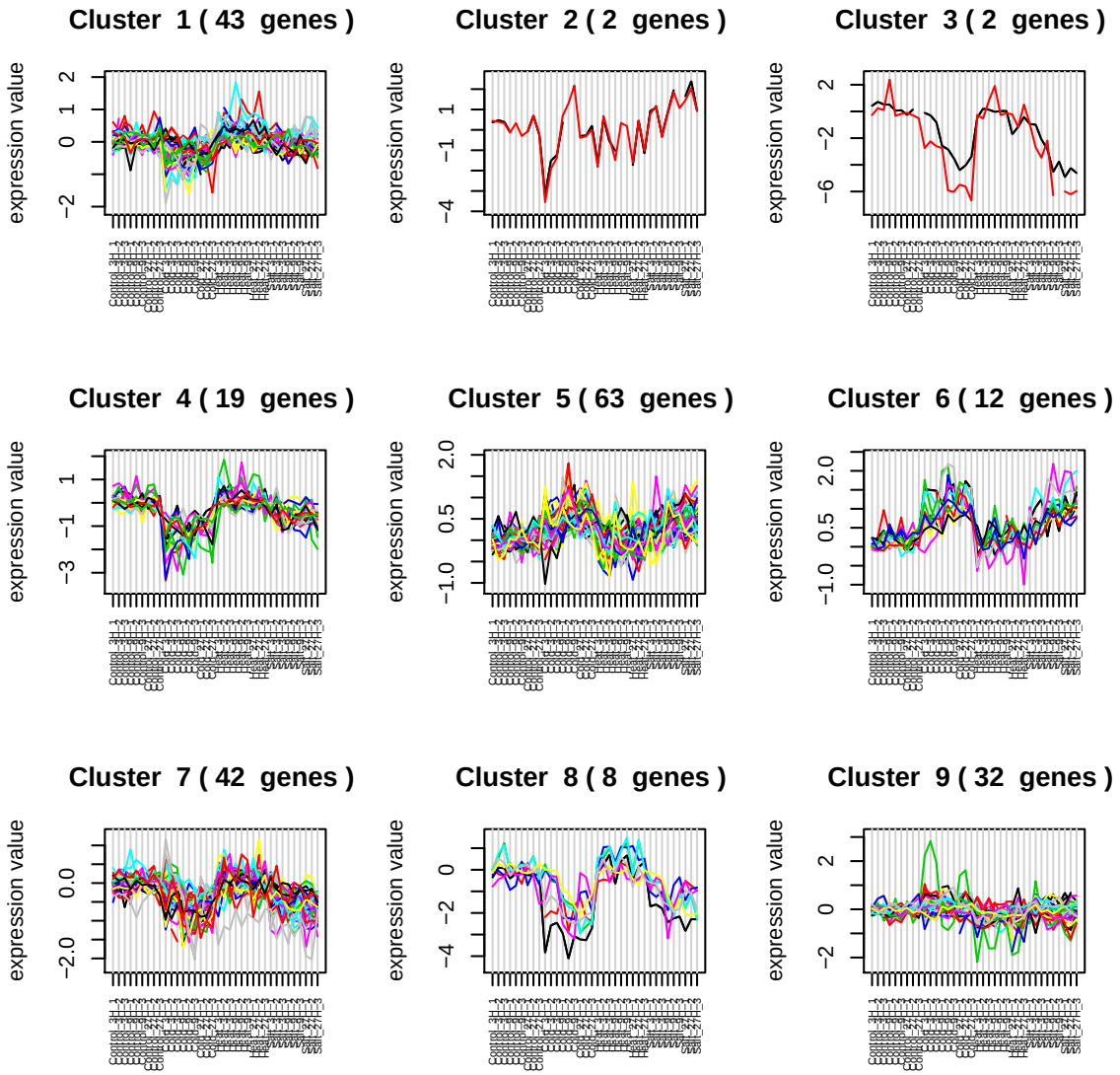


Figure 4: Cluster Analysis ColdvsControl significant genes

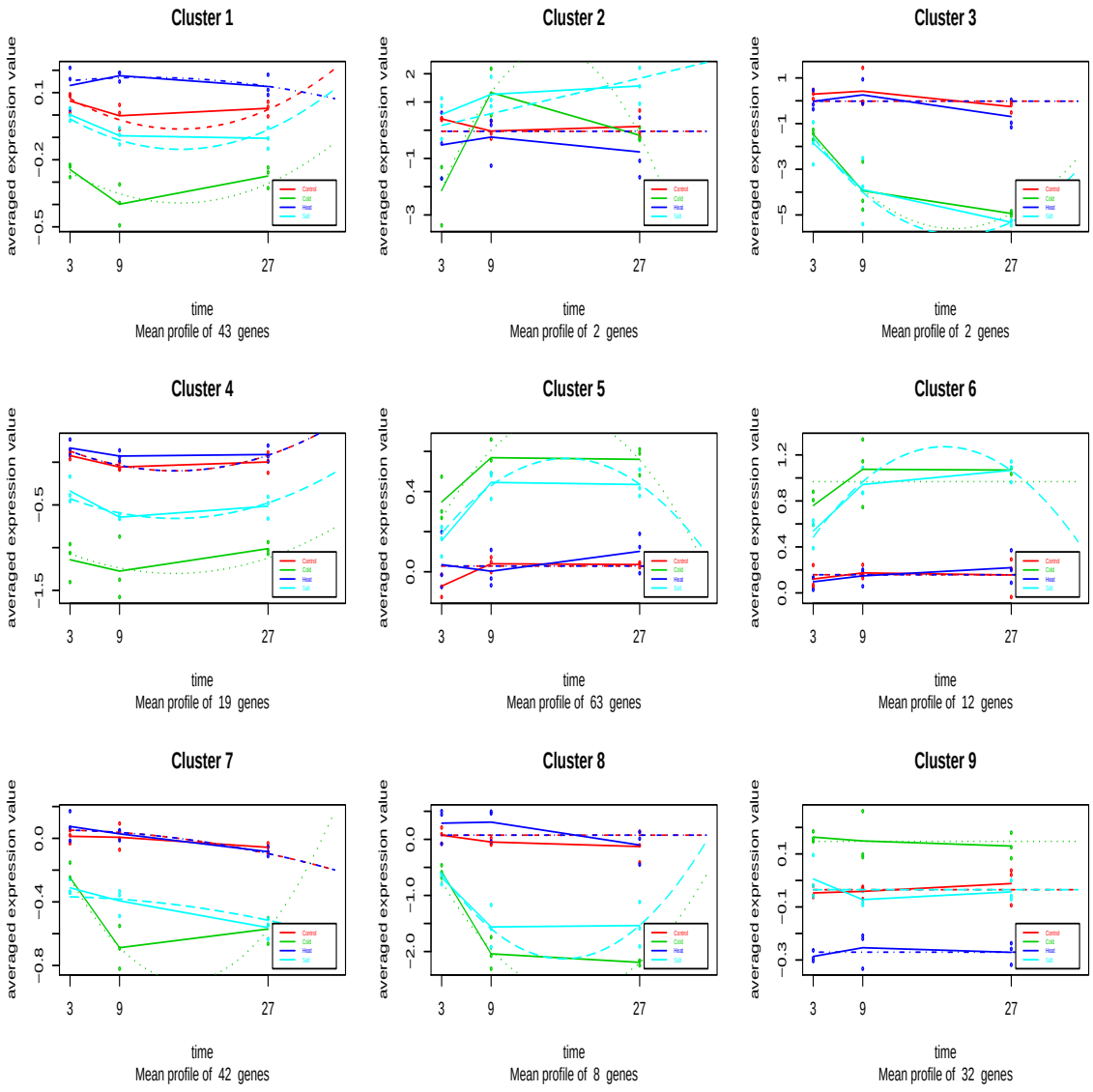


Figure 5: Expression Profiles ColdvsControl significant genes

```
  tc.dat <- NULL
  for (i in 1:n) {
    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)
```

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 aswell as the independent variable.
The following example illustrates to 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)
```

4.4 Long Time Series

The methodology described in this tutorial works fine with short time series experiments (up to 4 or 5 time points) where biologically meaningful responses can be modelled with polynomes up to a degree of 3. When a large number of time points is evaluated, an alternative to polynomials models with high degree on the time is the piecewise regression or splines regression. In this approach the time points can be split in two or more sets and in each set a polynomial with low degree can be fitted (6). To consider these models new dummy variables must be included in the model, now the dummies will differenciate between groups defined by the time, (in the example, d1=1 if t>t*).

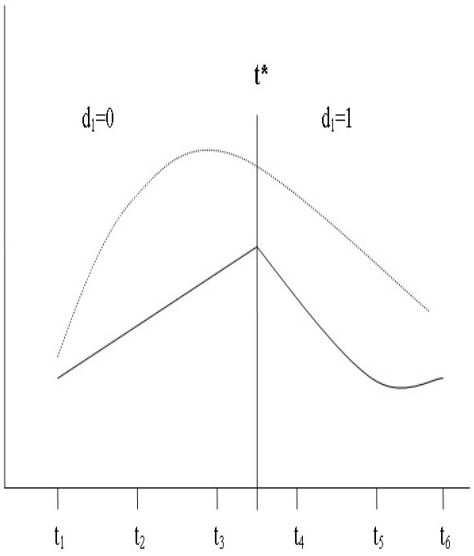


Figure 6: Dummy definition by splines regression

References

A. Conesa, M. Nueda, A. Ferrer, and M. Talón. masigpro: a method to identify significant differential expression profiles in time course microarray experiments., 2005. URL `http://www.ivia.es/centrogenomica/bioinformatics.htm`.

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*, 2005. URL `http://www.tigr.org/tdb/potato/index.shtml`.