Linear model for designed multivariate experiment (lmdme) - Vignette

Cristobal Fresno[1,2] and Elmer A. Fernández[1,2]

[1]Bio-science Data Mining Group, Universidad Católica de Córdoba
[2]CONICET, Argentina

October 2, 2012

1 Introduction

Current "omics" experiments (proteomics, transcriptomics, metabolomics or genomics) are multivariate by nature. Modern technology allows the exploration of the whole genome or a big subset of the proteome, where each gene/protein is in essence a variable explored to elucidate their relationship with some outcome. In addition, these experiments are including more experimental factors (time, dose, etc.) from design or subject specific information such as age, gender, linage and so on. Hence, in order to discover or evaluate experimental design or subject specific patterns, some multivariate approaches should be applied. In this context, Principal Component Analysis (PCA) and Partial Least Squares (PLS) are the most common. However, it is known that working with raw data could mask information of interest. So, ANOVA based decomposition is becoming popular to split variability sources, before the application of such multivariate approaches. Seminal works on genomics were "de Haan" et al. 2007 on APCA and Smilde et al. 2005 on ASCA models. However, as far as the authors know, R implementation of APCA is only available for spect data (ChemoSpec by Hanson 2012). Meanwhile, ASCA is only offered through a translation of the original Matlab© code (Nueda et al. 2007). But, the later only accepts up to three design matrices, limiting and making its use difficult. Here we provide a flexible implementation of "potentially" any linear model specification for ANOVA decomposition. It also provides both PCA or PLS analysis capabilities, statistical significance, parallel permutation test and graphical representations. The implementation is well-suited to directly analyze gene expression matrices from microarray and RNA-seq experiments.

2 The model

A detailed explanation of ANOVA decomposition and multivariate analysis can be found in Zwanenburg et al. 2011. Briefly, let's assume that G genes will be explored in a gene expression experiment (e.g. microarray or RNA-seq) with two main factors A with C levels $(A_1, \ldots, A_C)$ and B with D levels $(B_1, \ldots, B_D)$

for "j" replicates for each A, B and $A \cdot B$ combination levels. This implies an X^{GxN} matrix where $N = C \cdot D \cdot J$. Then, the ANOVA model for each gene can be written as (1):

$$x_{cdj} = \mu + \alpha_c + \beta_d + \alpha_c \cdot \beta_d + \varepsilon_{cdj} \tag{1}$$

where x_{cdj} is the measured expression for "some" gene, at combination "cd" of factors A and B for replicate "j"; μ is the overall mean; α, β and $\alpha \cdot \beta$ are the main and interaction effects respectively; and the error term $\varepsilon \sim N(0, \sigma^2)$. Equation (1) can also be expressed in matrix form for all genes (2):

$$X = \mu 1' + X_\alpha + X_\beta + X_{\alpha\beta} + E \tag{2}$$

where X_s matrices are of dimension GxN, μ and 1 are vectors of dimension $Gx1$ and $Nx1$ respectively. Matrices X_α, X_β, and $X_{\alpha\beta}$ contain the level averages for factors A, B and interaction between the two factors respectively.

2.1 The decomposition algorithm

Equation (2) is decomposed iteratively, where for each step a term is calculated and subtracted from the preceding residuals, to feed the next model as depicted in equation (3), where "$\wedge$" denotes estimated coefficients. In de Haan et al. 2007, Smilde et al. 2005 and Nueda et al. 2007 this procedure is based on mean calculations given the measurement position through design matrices, which are error prone. These matrices contain "1" or "0" to identify which measured, belongs or not to which factor respectively. On the contrary, in this library, the means are estimated by a maximum likelihood method using `lmFit` function provided by "*limma*" package (Smith 2003). Hence, statistical significance tests are automatically provided and, if required, empirical Bayes corrections can also be achieved.

$$\begin{array}{ll} step\ 0: & X = \mu 1' + E_\mu \hookleftarrow \\ step\ 1: & \rightarrow \hat{E}_\mu = X - \hat{\mu} 1' = X_\alpha + E_{\alpha\hookleftarrow} \\ step\ 2: & \rightarrow \hat{E}_\alpha = \hat{E}_\mu - \hat{X}_\alpha = X_\beta + E_{\beta\hookleftarrow} \\ step\ 3: & \rightarrow \hat{E}_\beta = \hat{E}_\alpha - \hat{X}_\beta = X_{\alpha\beta} + E_{\alpha\beta\hookleftarrow} \\ step\ 4: & \rightarrow \hat{E}_{\alpha\beta} = \hat{E}_\beta - \hat{X}_{\alpha\beta} = X_{\alpha\beta} + E_{\alpha\beta\hookleftarrow} \\ & E_k \sim N(0, \sigma^2), \quad k = \mu, \alpha, \beta, \alpha\beta \end{array} \tag{3}$$

2.2 PCA and PLSR analysis

Once the model is decomposed, PCA or PLSR can be carried out over each model term. In this context, PCA is concerned with explaining the variance structure of a set of observations (e.g. genes) through few linear combinations of variables (e.g. experimental conditions). It usually follows two main objectives: i) data reduction and ii) interpretation. When it is applied over residuals $\hat{E}_k$ with $k = \mu, \alpha, \beta, \alpha\beta$ it is known as APCA (de Haan et al. 2007), whereas when applied over coefficients $\hat{X}_k$ it is known as ASCA (Smilde et al. 2005). On the other hand, PLSR not only generalizes, but also combines features from regression and PCA, to deal with correlated explanatory variables in linear models (Abdi & Williams 2003, Shawe-Taylor & Cristianini 2005). It is particularly useful when one or several dependent variables (outputs - O) must be predicted from a large and potentially highly correlated set of independent variables (inputs X). In our case scenario, X could be the coefficient $\hat{X}_k$ matrix or the

residual $\hat{E}_k$ and the O matrix a diagonal or design information matrix when using the coefficients or residuals respectively; or even some particular user-defined matrix, such as a class matrix from Gene Ontology like in Gene Set Enrichment Analysis (GSEA) by Subramanian et al. 2005. In any case, the PLSR approach is useful to explore co-variability between gene expression and a predefined output class.

3 Other functionalities

- **Flexible input data**: just a GxN matrix, which is the typical data for gene/protein expression experiments, and a data.frame with the experimental design.
- **Statistics:** Student and F-test over coefficients are provided as well as leverage on PCA. They can be used to filter out rows/genes in PCA/PLSR analysis.
- **Permutation test:** a parallel permutation test implementation is also provided.
- **Visualization:** the package also offers different methods to visualize the results e.g. screeplots for PCA and biplots or loading plots for both PCA and PLSR.

4 Example

Prado-Lopez et al. 2010 studied differentiation of human embryonic stem cells under hypoxia conditions (Gene Expression Omnibus accession GSE37761). They measured gene expression at different time points for controlled oxygen levels. In this context, factor A stands for "*time*" with $C = 3$ levels $\{0.5, 1, 5 days\}$ and factor B for "*oxygen*" with $D = 3$ levels $\{1, 5, 21\%\}$ and $J = 2$ replicates, yielding a total of 18 samples. The rest of the dataset was excluded in order to account for balance design using the following commands:

```
> library(stemHypoxia)
> data(stemHypoxia) #This will load M and design objects in memory
> timeIndex   <- design$time %in% c("0.5","1","5")  #time levels
> oxygenIndex <- design$oxygen %in% c("1","5","21") #oxygen levels
> design<-design[ timeIndex & oxygenIndex,]# Both time & oxygen
> design$time  <-as.factor(design$time)
> design$oxygen<-as.factor(design$oxygen)
> rownames(M)<-M[,1] #Gene ID as row.names of M
> M <- M[,colnames(M) %in% design$samplename] #Just what is needed
```

Now we can explore microarray gene expression data present on `M` matrix, with N =40736 rows (genes) and 18 columns (samples/microarrays). In addition, the experimental `design` data.frame contains columns (main effects e.g. *time* and *oxygen*) and the sample names (`samplename`). Just to give an idea, the head of `design` and `M` (fot the first three microarrays) might look like these:

```
> head(design)
```

```
  time oxygen samplename
3  0.5      1    12h_1_1
4  0.5      1    12h_1_2
5  0.5      5    12h_5_1
6  0.5      5    12h_5_2
7  0.5     21   12h_21_1
8  0.5     21   12h_21_2
```

```
> head(M)[,1:3]
```

```
               12h_1_1   12h_1_2   12h_5_1
A_24_P66027   7.182159  7.511787  8.225355
A_32_P77178   6.385337  6.035340  6.440119
A_23_P212522  9.562124  9.390391  9.211380
A_24_P934473  6.287920  6.397256  6.264863
A_24_P9671   12.007126 11.995345 12.281969
A_32_P29551  10.175562  9.272561  9.360349
```

Then, the ANOVA decomposition (see section 2.1) of equation (1) can be obtained by:

```
> library(lmdme)
> fit <- lmdme(model=~time*oxygen,data=M,design=design)
> fit

lmDME object:
Data dimension:  40736  x  18
Design (head):
  time oxygen samplename
3  0.5      1    12h_1_1
4  0.5      1    12h_1_2
5  0.5      5    12h_5_1
6  0.5      5    12h_5_2
7  0.5     21   12h_21_1
8  0.5     21   12h_21_2

Model:~time * oxygen
Model deflaction:
  Step       Names            Formula CoefCols
1    1 (Intercept)                ~ 1        1
2    2        time        ~ -1 + time        3
3    3      oxygen      ~ -1 + oxygen        3
4    4 time:oxygen ~ -1 + time:oxygen        9
```

where a brief description of the used *data* and *design* information is displayed and how the decomposition process was carried out: which `Formula` was applied in the corresponding `Step`, how many coefficients where calculated for each gene (`CoefCols`) and how the steps were named (`Names`).
So, now, let's choose those subjects/genes where at least one interaction coefficient is statistically different from zero (F-test over the coefficients) and perform ASCA over them.

```
> id<-F.p.values(fit,term="time:oxygen")[[1]]<0.001
> sum(id) #The amount of genes for further exploration

[1] 305

> decomposition(fit,decomposition="pca", type="coefficient",
+               term="time:oxygen", subset=id, scale="row")
> biplot(fit,xlabs=rep("o",sum(id)), mfcol=NULL)
```

These instructions will perform ASCA and store the results inside the `fit` object. The user can also visualize the associated `biplot` (see Figure 1).

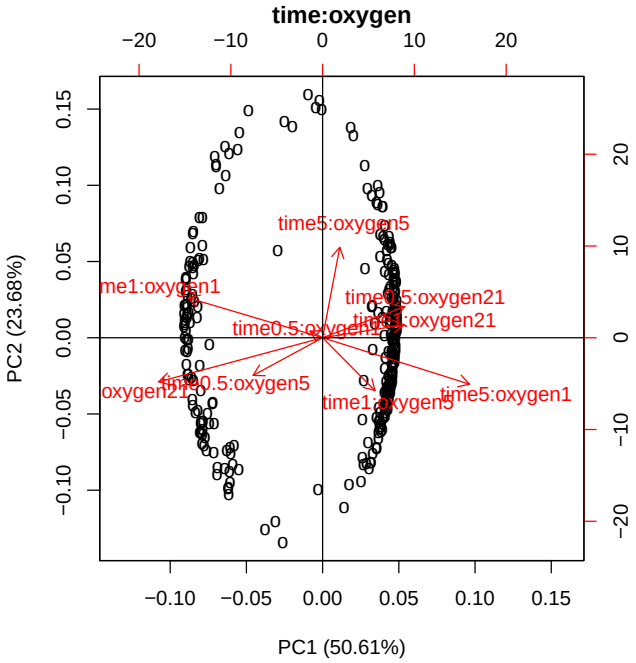

Figure 1: Biplot of ANOVA Simultaneous Component Analysis over genes satisfying F-value<0.001 over the interaction coefficients (time*oxygen)

In addition, PLSR can be applied on the same term, against the identity matrix (default option) and obtain the corresponding biplot (see Fig. 2).

```
> fit.plsr<-fit
> decomposition(fit.plsr,decomposition="plsr", type="coefficient",
+               term="time:oxygen", subset=id,scale="row")
> biplot(fit.plsr, which = "loadings", xlabs=rep("o",sum(id)),
+       ylabs=colnames(coefficients(fit.plsr,term="time:oxygen")[[1]]),
+        var.axes=TRUE, mfcol=NULL)
```

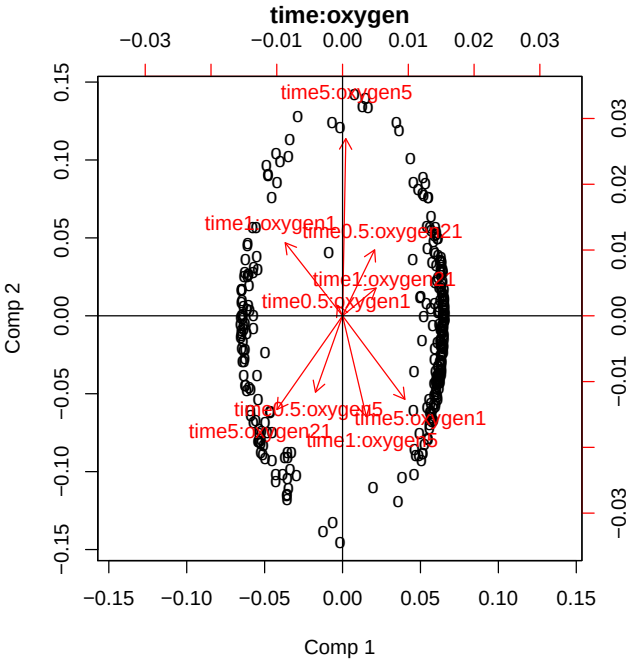

Figure 2: Biplot of ANOVA Partial Least Squares over genes satisfying F-value < 0.001 over the interaction coefficients (time*oxygen).

The interaction effect can also be displayed by means of using loadingplot function (see Fig. 3). In the case of an ANOVA-PCA/PLS analysis, the user only needs to change the `type="residuals"` parameter in `decomposition` function call.

```
> loadingplot(fit,term.x="time",term.y="oxygen")
```

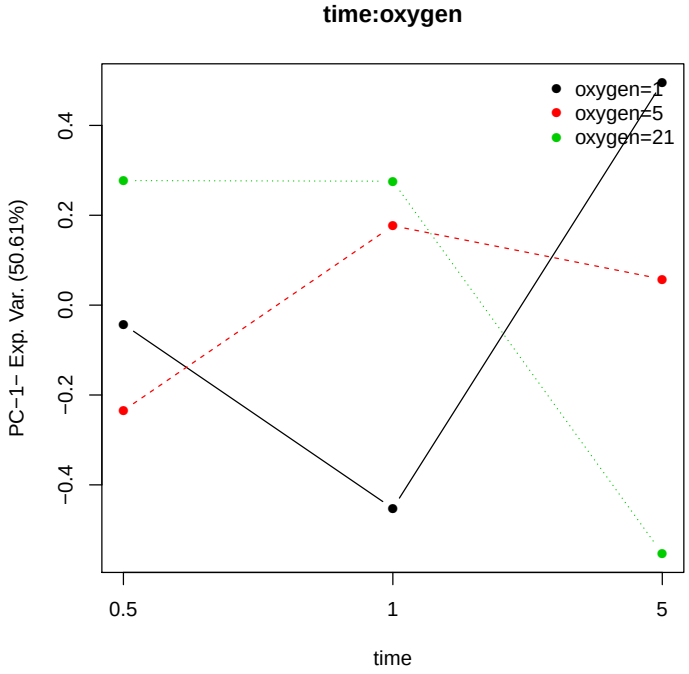

Figure 3: ANOVA Simultaneous Component Analysis loadingplot over genes satisfying F-value < 0.001 over the interaction coefficients (time*oxygen).

References

[1] Hertz,J. Krogh,A. and Palmer,R.G (1990) *Introduction to the theory of neural computation*, Westview Press, Oxford, USA.

[2] Abdi H, Williams LJ (2010) Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics* **2**, 433-459

[3] De Haan JR, Wehrens R, Bauerschmidt S, Piek E, van Schaik RC, Buydens LMC (2007) Interpretation of ANOVA models for microarray data using PCA. *Bioinformatics* **23**, 184-190.

[4] Hanson BA (2012) ChemoSpec: Exploratory Chemometrics for Spectroscopy. *package version* 1.51-2, academic.depauw.edu/∼hanson/ChemoSpec/ChemoSpec.html

[5] Nueda MJ, Conesa A, Westerhuis JA, Hoefsloot HC, Smilde AK, Talon M, Ferrer A. (2007) Discovering gene expression patterns in time course microarray experiments by ANOVA-SCA. *Bioinformatics* **23**, 1792-800.

[6] Prado-Lopez S, Conesa A, Arminan A, Martinez-Losa M, Escobedo-Lucea C, Gandia C, Tarazona S, Melguizo D, Blesa D, Montaner D, Sanz-Gonzalez S, Sepulveda P, Gotz S, O'Connor JE, Moreno R, Dopazo J, Burks DJ, Stojkovic M (2010) Hypoxia Promotes Efficient Differentiation of Human Embryonic Stem CellsFunctional Endothelium. *Stem Cells* **28**, 407-418.

[7] Shawe-Taylor J, Cristianini N (2005) Kernel Methods for Pattern Analysis. *Cambridge University Press*, ISBN 9780521813976

[8] Smilde AK, Jansen JJ, Hoefsloot HCJ, Lamer RAN, Van der Greef J, Timmerman ME (2005) ANOVA-simultaneous component analysis (ASCA): a new tool for analyzing designed metabolomics data. *Bioinformatics* **21**, 3043-3048

[9] Smyth GK (2004) Linear models and empirical bayes methods for assessing differential expression in microarray experiments. *Stat. Appl. Genet. Mol. Biol.* (**Article 3**).

[10] Subramanian A, Tamayo P, Mootha VK, Mukherjee S, Ebert BL, Gillette MA, Paulovich A, Pomeroy SL, Golub TR, Lander ES, Mesirov JP (2005) Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *PNAS* **102**, 15545-15550.

[11] Zwanenburg G, Hoefsloot HCJ, Westerhuis JA, Jansen JJ, Smilde AK (2011) ANOVA-principal component analysis and ANOVA-simultaneous component analysis: a comparison. *J. Chemometrics* **25**, 561-567

Session Info

```
> sessionInfo()

R version 2.15.1 (2012-06-22)
Platform: i386-pc-mingw32/i386 (32-bit)

locale:
[1] LC_COLLATE=C
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] stemHypoxia_0.99.1 lmdme_1.0.0        pls_2.3-0          limma_3.14.0
```

```
loaded via a namespace (and not attached):
[1] tools_2.15.1
```