Introduction to the MetStaT package

Gooitzen Zwanenburg
Tim Dorscheidt

Version 1.0

- 2013 -

Metstat

1.1 Introduction

MetStaT is a somewhat eclectic collection of metabolomics related tools bound together in a single R-package. The tools mostly originate with the Biosystems Data Analysis group at the University of Amsterdam (www.bdagroup.nl). Some were developed with support from the Netherlands Metabolomics Centre (NMC). Also the creation of this package was done with support from the NMC. The tools in this package include

- ASCA, a multivariate data exploration tool for designed experiments based on ANOVA decomposition followed by principal component analysis (PCA) [6].
- FoM, Figures of Merit is based on software written by Marinus van Batenburg to calculate parameters that can be used to quantify the quality of measured data [1].
- RevNet gives three methods that can be used to reverse engineer metabolomic networks from measured concentration data [3]. The software is based on the corresponding Matlab code written by Diana Hendrickx.
- QStat gives an implementation of Goeman's global test to determine whether a group of genes has different expression patterns under different conditions [4].
- PLSDA2CV, double cross validated PLSDA analysis.

1.2 Figures of Merit

In the field of metabolomics it is not uncommon to measure concentrations of hundreds of metabolites simultaneously using a variety of platforms such as chromatography platforms coupled with mass spectrometry or through NMR spectrometry. The repeatability of measurements can be taken as a figure of merit when experimental conditions between measurements can be kept the same with a high degree of confidence. So, the same analyst performs the measurements using the same procedures on the same equipment. When experiments involve large numbers of samples measured in different batches, as is frequently the practice in omics studies, additional sources of error are introduced and errors can no longer be assumed to be constant and uncorrelated. In their paper Van Batenburg et

al. [1] introduce two new figures of merit that take into account the correlation between metabolites and difference in variance structure between batches. The tool included in this package calculates the measurement error distribution for a single variable (metabolite).

The measurement variance of a metabolite, σ_s^2, is estimated from repeated analysis of a sample s. The results are assumed to be normally distributed, $N(\mu_s, \sigma_s^2)$, with μ_s the population mean. In omics studies the number of repeated measurements of a sample is mostly too limited to obtain a estimate of the variance at all levels of signal intensity, μ_s. Therefore the following estimation procedure is used to estimate the variance as a function of the signal intensity. Estimated values $\hat{\sigma}_s$ and $\hat{\mu}_s$ are sorted according to the intensity $\hat{\mu}_s$. The sorted $(\hat{\mu}_s, \hat{\sigma}_s)$ pairs are then grouped into bins of N $\hat{\sigma}_s$ levels.

The measurement variance, σ_s^2 is then modeled by the Rocke-Lorenzato model:

$$\sigma_s^2 = \sigma_{\text{Add}}^2 \quad \text{if} \quad \mu \leq \alpha_{\text{Mult}} \tag{1.1}$$

$$\sigma_s^2 = \sigma_{\text{Add}}^2 + \sigma_{\text{Mult}}^2(\mu - \alpha_{\text{Mult}})^2 \quad \text{if} \quad \mu > \alpha_{\text{Mult}} \tag{1.2}$$

The value of the measurement variance is determined by two ranges. For intensities below the threshold α_{Mult} the variance is a constant σ_{Add}^2, the measurement variance in absence of metabolite intensity, i.e. only background noise is present. This parameter can be estimated from blank samples. For intensities above the threshold α_{Mult}, an intensity dependent variance with proportionality constant σ_{Mult}^2 also contributes to the measurement variance and the variance increases with increasing intensity.

The function `FoM.Calculate` takes the data for a single variable (metabolite), orders and bins the data by intensity and determines the best fit with the Rocke-Lorenzato model. The function returns the parameters for the best fit, α_{Mult} and σ_{Mult} and the sum of the squared residuals. The fit to the Rocke-Lorenzato model is done by fitting the parameters σ_{Mult} and σ_{Add} to measurement data for varying values of α_{Mult}. In this way the best parameters fitting the model of equations (1.1) and (1.2) are found.

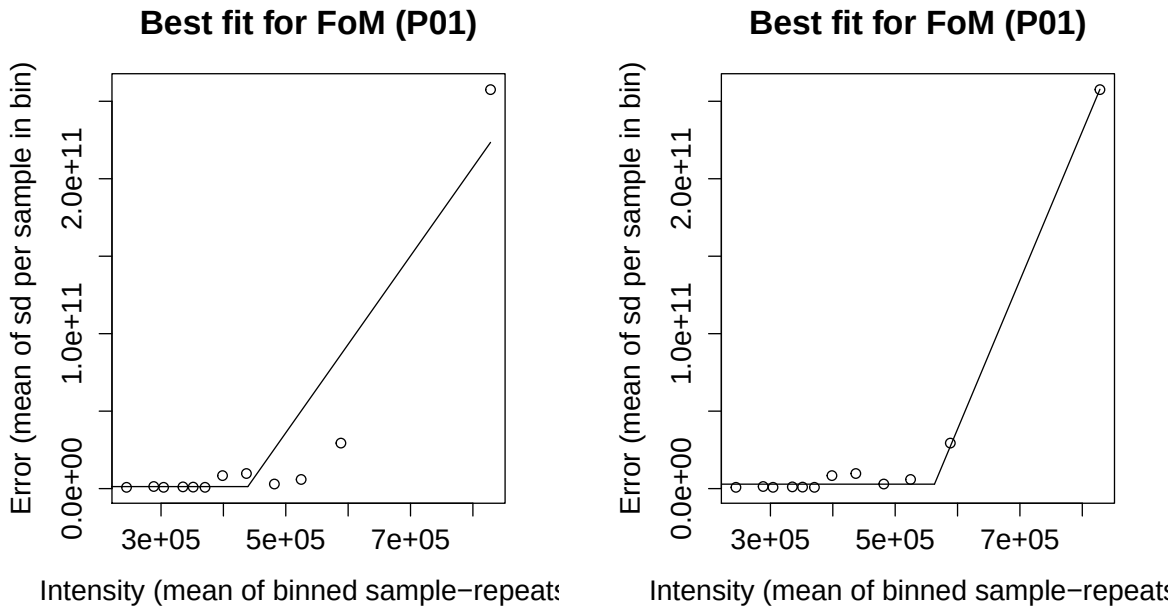
1.2.1 Example

The example included in the package consists of a data matrix with three columns. The first two columns are used to identify the samples, the third column contains the measured values. The data could represent concentrations of a plant metabolite collected in different batches (numbered in the second column) from (in this example 15) different plots in a test field. The program can handle only a single variable. If more variables (metabolites) are present in the data matrix then they have to be analyzed one by one. A simple script running through the columns might be handy if the number of variables is large. A typical run would be:

```
data(FoMData)
Res <- FoM.Calculate(FoMData, cols.id = c(1,2), 3, 5, quiet=FALSE,
+ repeats.per.id = -1, fit.type = 2, alpha.steps = 100)
```

The identifying columns (`Plot` and `Batch`) are given by `cols.id = c(1,2)`, the third argument gives the data column, the number of rows per bin is here taken as 5. Verbose

output is given by setting `quiet=FALSE`, with `repeats.per.id = -1` we do not restrict the number of rows per bin. For the fitting of the data to the Rocke-Lorenzato model either the default `lm` method is used (`fit.type=1`) or the more robust `rlm` (`fit.type=2`) from the `MASS` package is used. The number of different values of α_{Mult} that are tried is determined by `alpha.steps`.



Figuur 1.1: Mean variance per bin versus mean intensity of the values in the bin for $\alpha_{\text{Mult}} = 4$ (left panel) and $\alpha_{\text{Mult}} = 100$ right panel. Clearly the lower value for α_{Mult} is too low as the cutoff from $\mu \leq \alpha_{\text{Mult}} to \mu > \alpha_{\text{Mult}}$ is at too low a value. Increasing the number of trial values of α_{Mult} improves the result.

The output gives for each value of α_{Mult} the parameters for the best fit with that particular value of α_{Mult}. The overall best fit values are given in `Res$best.fit`. In figure 1.1 the results are shown for two values of `alpha.steps`. The lower one (left panel) 4, is clearly too low. The higher value of 10 (right panel) gives a better result. This is also reflected in the sum of squares of the residuals:

```
$best.fit
                                [,1]
alpha                    5.689049e+05
additiveCoefficient      2.865141e+09
multiplicativeCoefficient 9.839764e+05
ssQresiduals             1.623603e+20
```

for `alpha.steps=4` and

```
$best.fit
                                [,1]
alpha                    5.630103e+05
```

```
additiveCoefficient        2.865141e+09
multiplicativeCoefficient  9.607982e+05
ssQresiduals               1.127972e+20
```

for `alpha.steps=100`

1.3 Metabolomic Networks

1.3.1 Reverse engineering of metabolomic networks

Several methods exist for the reconstruction of metabolic networks from metabolite concentration data. Three such methods have been implemented in this package. A short description follows, for more details see [3] and references therein. The time-lagged correlation method looks at correlations between time series that are time shifted with respect to each other. The idea being that the influence of one species on another occurs only after a time delay. Time-lagged correlations between time profiles of metabolite concentrations that exceed a set threshold are considered to have a direct interaction. The size of the threshold has to be determined independently, for example form the fraction of real interactions among the possible interactions in real networks of a specific size.

The Jacobian method considers the kinetic model for the metabolomic network:

$$\frac{\mathrm{d}S_i(t)}{dt} = F_i(S_1(t), \cdots, S_n(t)) \tag{1.3}$$

where the $S_i(t)$ is the concentration profile of metabolite i. If metabolite i influences metabolite j, the element

$$\frac{\partial F_j}{\partial S_i} \tag{1.4}$$

of the Jacobian matrix is not zero, whereas the element in the Jacobian matrix will be zero if metabolite i has no effect on metabolite j. Metabolomic networks are sparse but the Jacobian in general is not. The number of edges is reduced by considering a linear approximation of the Jacobian in which the concentrations S_i are only slightly different from the equilibrium concentrations. The Jacobian can then be estimated from a constrained least squares estimation, where the constraints serve to get a sparse result. By setting a cutoff value for the elements of the Jacobian matrix below which no interaction between elements i and j is assumed, the non-zero elements of the Jacobian matrix can be used to construct the metabolic network.

A third method, somewhat related to the Jacobian method, is the zero slopes method. Here the effect of changing the concentration of a single metabolite on the concentration profiles of the remaining metabolites is measured. Depending on the response of metabolite j to a change in metabolite i (all other metabolite concentrations remaining constant), inferences on the connection between metabolites i and j are made.

1.3.2 Examples

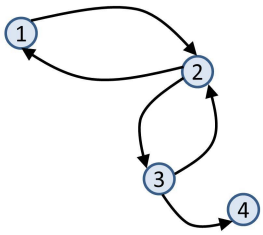
The Jacobian method is demonstrated with an example comprising measured concentration profiles for four metabolites for four different experiments. In each of the experiments one of the metabolites' concentration is raised by 2% from the steady state level. The concentrations are measured at 51 time points. Note that the algorithm expects the time intervals between data points to be the same. The data matrix in the example has dimensions $51 \times 4 \times 4$. The slice $[,,1]$ contains data from the first experiment in which the concentration of the first metabolite was increased with 2% over the steady state value. The columns of the slice represent the four metabolites, each row the measured concentrations at different times. The time interval between measurements is 0.01 min. The penalties for the constrained least squares are set to `lambda.penalty = 0.0002` and `kappa.penalty = 1`. The threshold for values of the Jacobian to be considered different from 0 is `jacobian.threshold = 0.0001`. Running the example gives:

```
data(RevNetJacobian)
RevNet.JacobianMethod(RevNetJacobian, 0.01, SteadyState, 0.0002, 1, 0.0001)
```

The result is the pruned Jacobian that represents the network: a 1 in a row/column combination means that the corresponding nodes are connected:

```
     [,1] [,2] [,3] [,4]
[1,]    1    1    0    0
[2,]    1    1    1    0
[3,]    0    1    1    0
[4,]    0    0    1    1
```

The directed graph derived from the Jacobian is shown in figure 1.2



Figuur 1.2: Directed graph derived from the Jacobian method to reverse engineer a metabolic network.

The time lagged method example matrix contains 100 time points (rows) at which the concentrations of four metabolites (columns) are measured. Starting at time $t = 0$ the concentration of metabolite 1 is increased by 20% every 9 seconds. The maximum time lag `max.time.lag = 55`. The threshold for this matrix was set at the 55-th quantile, `threshold=.55`. Running this example gives the following network.

```
> data(RevNetTimeLagged)
> RevNet.TimeLaggedMethod(RevNetTimeLagged, 50, .55)
```

```
      [,1] [,2] [,3] [,4]
[1,]    1    1    0    0
[2,]    1    1    1    0
[3,]    0    1    1    1
[4,]    0    0    1    1
```

The data matrix to demonstrate the zero slopes method is not unlike the matrix for the Jacobian but the experimental conditions are different. In the four experiments the concentration of one of the four metabolites is increased by 20%. The zero slopes method determines the effect of this change on the other metabolites. Running the example gives the following output:

```
> data(RevNetZeroSlopes)
> RevNet.ZeroSlopesMethod(RevNetZeroSlopes, 0.001, 0.00005)
      [,1] [,2] [,3] [,4]
[1,]   -1    1    0    0
[2,]    1   -1    1    0
[3,]    0    1   -1    0
[4,]    0    0    1   -1
```

In this example the time between measurements is `deltat=0.001` minutes and values smaller than `threshold=0.00005` are taken as zeros.

1.3.3 Goeman's global test Q-statistic

Goeman's test [2] is used to detect significant differences between conditions (for example treatment vs. placebo) at the group level [4]. Groups may include metabolites that are produced in certain biological pathways or groups of genes that are expressed together. This contrasts with methods to detect differences at the level of individual genes or biomarkers. Goeman's global test uses a test statistic Q to evaluate the significance of differences between the groups. The function `QStat.Calculate` calculates the Q statistic and does a permutation test to obtain the p-value. A sample run with the data supplied by the package:

```
data(QStat)
QStat.Calculate(QStatX, QStatY, 1000)
[1] "Performing test using 1000 random permutations."
$y.boolean
      [,1]
 [1,] TRUE
 [2,] TRUE
....
[16,]  1.25004909  0.27568869 -0.974640062  0.88497120 -0.66263523  0.158295994
[17,]  0.51953970 -0.24632704 -0.594107855  0.95157585  0.11064022  0.127053416
[18,]  0.49959837  0.17683026  0.075268688  0.30903975  1.44653680  0.007720738
$Q
         [,1]
[1,] 39.77629
```

```
$p
[1] 0.02792632
```

returns the original data and the binary vector $\mathbf{Y}$ that determines which rows are included in the group. The printing of the data and $\mathbf{Y}$ is suppressed in the output example above. Note that preprocessing of the data, like auto scaling, needs to be done by the user. Here, the program is called with a data matrix $\mathbf{X}$ with 6 columns corresponding to 6 variables, and 18 rows. The data represent an experiment with two conditions. The first 10 rows are measured under condition 1, the last 8 rows are measured under condition 2.

1.4 ASCA

ASCA is a multivariate form of ANOVA. The general idea of ASCA is to decompose the data matrix, $\mathbf{X}$, ANOVA style in a sum of mutual orthogonal effect matrices, $\mathbf{X}_\alpha$, $\mathbf{X}_\beta, \cdots$ that describe the main experimental factors and matrices $\mathbf{X}_{(\alpha\beta)}, \cdots$ describing the interactions between the main factors [5, 6]:

$$\mathbf{X} = \mathbf{1}^{\mathrm{T}}\mathbf{X} + \mathbf{X}_\alpha + \mathbf{X}_\beta + \cdots + \mathbf{X}_{(\alpha\beta)} + \cdots + \mathbf{E} \tag{1.5}$$

The first term on the right hand side is the overall mean, $\mathbf{1}^{\mathrm{T}}$ is a row matrix in which each element is 1. The matrices $\mathbf{X}_\alpha, \mathbf{X}_\beta, \cdots$ describe the main experimental factors, the two-factor interactions are described by $\mathbf{X}_{(\alpha\beta)}, \cdots$. The last term, $\mathbf{E}$ contains the residuals, variation in the data that cannot be described by the effect and interaction matrices. The data matrix for ASCA is assumed to be balanced, i.e. the number of measurements is the same for each treatment level. The effect matrices contain the level averages for each of the treatment levels. ASCA does a PCA on these matrices and thus seeks out the direction of largest variance in the level averages of the effect matrices.

The package main function to do ASCA is `ASCA.Calculate`. This function takes the data matrix and the design matrix as its first two arguments. The factors and interactions that need to be calculated can be entered as additional arguments. There is also the option to auto scale the data. The function returns the variances explained by the different terms in equation (1.5) together with a the factor matrices and matrices resulting from a singular value decomposition from which scores and loadings can be retrieved. The list can be used in the function `ASCA.DoPermutationTest` to perform a permutation test to establish the significance of the experimental factors and interactions [7].

1.4.1 Example

The use of ASCA is illustrated with an example data matrix, included in the package, with four dependent variables and two experimental factors. The first factor α has two treatment levels, the second factor β has three treatment levels. Each observation is repeated 10 times, hence the total number of measurements in the data matrix is $2 \times 3 \times 10 = 60$. The experimental design is represented by the design matrix $\mathbf{F}$. Each column of the

design matrix corresponds to an experimental factor, $\alpha, \beta, \cdots$, each row corresponds to a measurement. The elements of each row indicate the treatment levels the measurement belongs to. For example:

```
> head(ASCAF, 12)
      [,1] [,2]
 [1,]    1    1
 [2,]    1    1
 [3,]    1    1
 [4,]    1    1
 [5,]    1    1
 [6,]    1    1
 [7,]    1    1
 [8,]    1    1
 [9,]    1    1
[10,]    1    1
[11,]    1    2
[12,]    1    2
```

We see that the first measurement falls treatment level 1 for the first factor and also in treatment level 1 of the second factor. This is true for the first 10 measurements (rows). The eleventh measurement is again in the first treatment level for the first factor, α, but in the second treatment level for the second factor β.

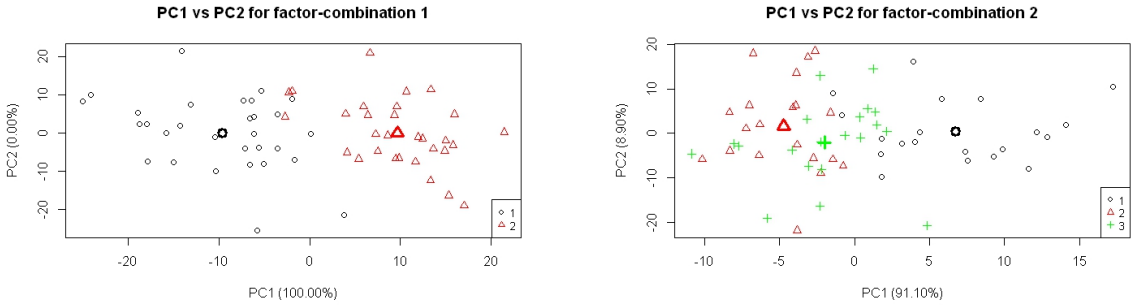
When we run the example with the data matrix `ASCAX` and design matrix `ASCAF` we receive the output list `ASCA` and statistics on the explained variance:

```
> data(ASCAdata)
> ASCA <- ASCA.Calculate(ASCAX, ASCAF, equation.elements = "1,2,12",
+ scaling = FALSE)
Variance explained per principal component (if >1%):
Whole data set  PC1: 51.53%   PC2: 32.27%   PC3: 16.03%
Factor 1        PC1: 100.00%  PC2:  NA%     PC3:  NA%
Factor 2        PC1: 91.10%   PC2: 8.90%    PC3:  NA%
Interaction 12  PC1: 92.22%   PC2: 7.78%    PC3:  NA%

Percentage each effect contributes to the total sum of squares:
Overall means   99.39%
Factor 1        0.19%
Factor 2        0.05%
Interaction 12  0.03%
Residuals       0.34%

Percentage each effect contributes to the sum of squares of the centered data:
Factor 1        31.11%
Factor 2        8.74%
Interaction 12  5.08%
Residuals       55.07%
```

We see that PCA on the complete data matrix gives an explained variance of 52% for the first principal component, 32% for the second principal component and 16% for the third principal component. The contribution of factor 1 is clearly dominant, the second factor and interactions contribute about the same to the sum of squares. It is also clear that a significant fraction of unexplained variance remains in the residuals. The output list `ASCA` can be used for plotting scores and loading plots, `ASCA.Plot(ASCA)`. In Figure 1.3 we show the scores plots for the two factors. The rank of the effect matrices is very low (equal



Figuur 1.3: Scores plots for the two factors of the data example. The large symbols represent the scores of the effect matrices, the corresponding smaller symbols are the projections of the data matrix onto the plane spanned by the first two principal components.

to the number of treatment levels minus 1). Thus, the first experimental factor only has a single principal component, explaining the 110% explained variance of the first principal component of the first factor. Because of the low rank, the scores plot of the effect matrices only has a few points. The scores plots shown in Figure 1.3 also include the projections of the data matrix onto the plane spanned by the first two principal components. In this way, a visual overview of the distribution of the measurements can be obtained.

A permutation test is available to calculate the p-values for the significance of the experimental effects:

```
> ASCA.DoPermutationTest(ASCA)
          1     2     12
[1,] 0.001 0.001 0.176
```

This shows that the two factors have a significant effect, but their interaction does not.

1.5 PLSDA Double Cross Validation

Bibliografie

[1] Marinus F. Van Batenburg, Leon Coulier adn Fred van Eeuwijk, Age K. Smilde, and Johan A. Westerhuis. New figures of merit for comprehensive funcional genomics data: The metabolomics case. *Anal. Chem.*, 83:3267 – 3274, 2011.

[2] J.J. Goeman, S.A. van de Geer, F. de Kort, and H.C. van Houwelingen. A global test for groups of genes: tesing association with a clinical outcome. *Bioinformatics*, pages 93 – 99, 2004.

[3] Diana M. Hendrickx, Margriet M.W.B. Hendriks, Paul H.C. Eilers, Age K. Smilde, and Huub C.J. Hoefsloot. Reverse engineering of metabolic networks, a critical assesment. *Mol. BioSyst.*, 7:511 – 520, 2011.

[4] Diana M. Hendrickx, Huub C.J. Hoefsloot, Margriet M.W.B. Hendriks, André Canelas, and Age K. Smilde. Global test for metabolic pathway differences between conditions. *Analytica Chimica Acta*, 719:8 – 15, 2012.

[5] Jeroen J. Jansen, Huub C.J. Hoefsloot, Jan van der Greef, Marieke E. Timmerman, Johan A. Westerhuis, and Age K. Smilde. Anova-simultaneous component analysis (asca): a new tool for analyzing designed metabolomics data. *Bioinformatics*, 19:469 – 481, 2005.

[6] Age K. Smilde, Jeroen J. Jansen, , Huub C.J. Hoefsloot, Robert-Jan A.N. Lamers, Jan van der Greef, and Marieke E. Timmerman. Anova-simultaneous component analysis (asca): a new tool for analyzing designed metabolomics data. *Bioinformatics*, 2:3043 – 3048, 2005.

[7] Gooitzen Zwanenburg, Huub C.J. Hoefsloot, Johan A. Westerhuis, Jeroen J. Jansen, and Age K. Smilde. Anova-principal component analysis and anova-simultaneous component analysis: a comparison. *Bioinformatics*, 25:561 – 567, 2011.