Efficient ANOVA Based Microarray Analysis using FSPMA

P. Sykacek
[1]Department of Biotechnology
BOKU University
peter@sykacek.net

Chapter 1

Introduction

This lecture attempts to connect statistical rigour with efficient analysis of microarray datasets. In order to ensure meaningful results, it is important to avoid wrong use of models and thus to understand the basics of the underlying statistical model. The minimum requirement for a succesful application of the tool discussed here is an understanding of effects (random and fixed effects), contrasts, multiple testing issues and a simple understanding of issues of experiment design.

To apply the information provided in this lecture in practice, it is vital to consider some of the technical settings of real world scenarios. Analysis is based on different sources of data (e.g. single vs. two channel arrays) and experiments provide ill specified datasets or missing information. A version update of the quantification tool for example often modifies the output file structure, without necessarily providing a tool to reformat old version datafiles. If that happens during an experiment (e.g. data collection of time course data of a developmental process can take several months), the data that should be analysed will show structural inconsistencies. Other problems arise from missing data (e.g. spots of poor quality were rejected during quantification) and technical artifacts that might affect data in a non random fashion.

Before we talk about a specific tool for microarray analysis which allows to approach most of these issues in an efficient manner, it is important to highlight an unavoidable consequence of data analysis. Results do not only depend on data. The method used for analysis has always some influence as well. Often this influence is non negligible. As an example, we might apply two quite common methods for "dimensionality reduction". Dimensionality reduction is a well known approach in, so called, exploratory data analysis. Figure 1 illustrates an application of principal component analysis (PCA) and linear discriminant analysis (LDA) to a labeled dataset. Both methods attempt to find a lower dimensional projection, that maximises the information captured by the new representation. PCA ignores the labels and maximises the variance in the lower dimensional space, whereas LDA maximises class separability. The data is to some extent pathological, since the maximum variance direction does not provide any information about class separability. In such cases, the directions found by the two methods will be orthogonal and the interpretation completely different. The take home message of this example is not to expect that there is a single objectively correct result. Analysis results depend on methods

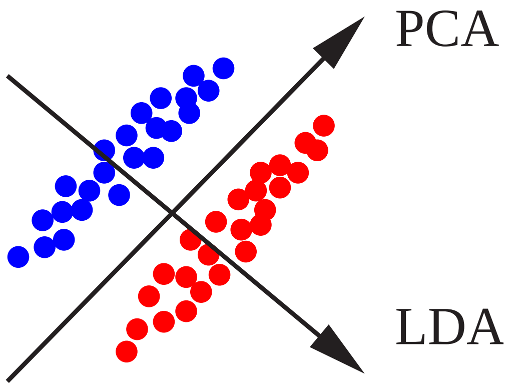


Figure 1.1: Dependency of dimensionality reduction on the method used to calculate the projection. Principal component analysis (PCA) favours large variance, whereas linear discriminant analysis (LDA) uses class information (i.e. red vs. blue) and maximises class separability. Each method provides thus a very distinct interpretation of the data. We obtain in this pathological case lower dimensional projections onto orthogonal directions.

and we are not neccesarily able to rule out all but one of the methods based on their properties. We will thus obtain different perceptions of data that are equally plausible and the "real truth" can only be reveled by follow up experiments.

We use in this lecture the freindly statistics toolbox for microarray analysis for efficient analyis of microarray experiments. The library is available under GNU GPL 2 license, it can be downloaded from `http://www.sykacek.net/software.html#fspma`. The underlying idea in FSPMA differs from most other R-packakges for microarray analysis, since in general, FSPMA does not require scripting to taylor analysis to a specific experiment. FSPMA is configured by a tab delimited text file which allows to taylor the library to any balanced experiment. This "definition file" is used to

- Specify the experimental layout.
- Assign quantified microarrays to experimental conditions.
- Assign the logical meaning of columns based on their names. (i.e. the user specifies which column name is used as gene symbol, colour channel, spot position etc.)
- Provide names of all gene ids that should be used in the analysis.
- Formulate potentialy several comparisons that should be used to provide rank lists.
- Specify normalisation and an "imputation" scheme. (The latter is used to fill in missing values).

The setup of the definition file is such that we can analyse poorly designed experiments with non thought through slide layout that possibly consist of heterogeneous datafiles. Although it is strictly recommended not to use such bad practices, dealing with these problems is unfortunately a common task for the practitioning bioinformatician.

Chapter 2

Theoretical ANOVA Essentials

2.1 ANOVA preliminaries

The mathematical relations of ANOVA models are most easily derived with the help of Lemma 2.1.1.

Lemma 2.1.1 (Residual sum of squares). *The residual sum of squares of an expression like Equation (2.1)*

$$SS = \sum_{i=1}^{N} (x_i - \mu)^2 \tag{2.1}$$

obtained from plugging in $\hat{\mu} = \underset{\mu}{argmin} \left(\sum_{i=1}^{N} (x_i - \mu)^2 \right)$ *can be expressed as*

$$RSS = \sum_i x_i^2 - \frac{1}{N} \left(\sum_i x_i \right)^2 . \tag{2.2}$$

Proof. We rewrite Equation (2.1) in vector notation.

$$SS = (\boldsymbol{x} - \mu\mathbf{1})^T (\boldsymbol{x} - \mu\mathbf{1}),$$

where $\mathbf{1}$ denotes a $[N \times 1]$ dimensional column vector. Setting the derivative $\nabla_\mu SS$ equal to zero we get

$$\hat{\mu} = \frac{\boldsymbol{x}^T\mathbf{1}}{\mathbf{1}^T\mathbf{1}}$$

and finally

$$RSS = \boldsymbol{x}^T\boldsymbol{x} - \frac{\boldsymbol{x}^T\mathbf{1}\boldsymbol{x}^T\mathbf{1}}{\mathbf{1}^T\mathbf{1}},$$

which is Equation (2.2) reexpressed in vector notation. □

2.2 One way ANOVA with fixed effects

A one way ANOVA with fixed effects assumes the model

$$y_{ij} = \mu + \alpha_i + \epsilon_{ij}, \tag{2.3}$$

where $i = 1, .., a$ and $j = 1, .., n$ denote the number of levels (classes) and the number of observations (samples) per class. We assume a balanced design, where the number of samples in each class is identical. Additional assumptions of the ANOVA model are

- The sum $\sum_i \alpha_i$ equals 0. This is a necessary identifiability constraint.
- The ϵ_{ij} are independent identically distributed samples from a zero mean Gaussian distribution with standard deviation σ (i.e. $\epsilon_{ij} \sim \mathcal{N}(0, \sigma)$).

The fixed effects ANOVA furthermore assumes that the α_i are parameters of an exhaustive list of levels with the null hypothesis that all α_i are zero. In a freqeuentist understanding the α_i are non random. The corresponding probabilistic assumption would be that each α_i is distributed according to it's own independent distribution i.e. $p(\boldsymbol{\alpha}) = \prod_i p(\alpha_i)$. Finally we note that this model is equivalent to a model with one mean $\mu_i = \mu + \alpha_i$ for each level and the null hypothesis that all μ_i are equivalent to μ.

2.2.1 Residual sums of squares

The fixed effects model allows to write three different residual sums of squares, which we may reexpress using Lemma 2.1.1.

- We have the Total Residual Sum of Squares

$$RSST = \sum_{i,j} (y_{ij} - \bar{y}_{..})^2 = \sum_{i,j} y_{ij}^2 - \frac{1}{na} \left(\sum_{i,j} y_{ij} \right)^2 \tag{2.4}$$

with degrees of freedom

$$dfT = na - 1 \tag{2.5}$$

- We have the Residual Sum of Squares between Levels

$$RSSB = \sum_i n(\bar{y}_{i.} - \bar{y}_{..})^2 = n \left(\sum_i \bar{y}_{i.}^2 - \frac{1}{a} \left(\sum_i \bar{y}_{i.} \right)^2 \right) \tag{2.6}$$

with degrees of freedom

$$dfB = a - 1 \tag{2.7}$$

- We also get the Residual Sum of Squares within Level

$$RSSW = \sum_{i,j} (y_{ij} - \bar{y}_{i.})^2 = \sum_i \left(\sum_j y_{ij}^2 - \frac{1}{n} \left(\sum_j y_{ij} \right)^2 \right) \tag{2.8}$$

with degrees of freedom

$$dfW = a(n - 1) \tag{2.9}$$

Sums of squares and fegrees of freedom are additive. We get

$$RSST = RSSW + RSSB \text{ and } dfT = dfW + dfB,$$

which we quote without proof. We also note that each residual sum of squares together with the corresponding degrees of freedom results in a mean square error ($MSET = RSST/dfT$, $MSEB = RSSB/dfB$ and $MSEW = RSSW/dfW$). We conclude this section with a lemma that is useful for the further analysis of this ANOVA model.

Lemma 2.2.1 (Estimation by $RSSB$ in fixed effects ANOVA). *The residual sum of squares $RSSB$ in Equation (2.6) estimates*

$$\begin{aligned} & \sigma^2(a-1) + n\sum_i(\mu_i - \mu)^2 \\ = \quad & \sigma^2(a-1) + n\sum_i \alpha_i^2 \end{aligned} \tag{2.10}$$

Proof. As proof, we investigate the expectation of the residual sum of squares between levels $\langle RSSB\rangle$, where the expectation is taken w.r.t. the "distribution" of the "estimates"[*] $\bar{y}_{i.}$ and $\bar{y}_{..}$. Using $\sum_i \bar{y}_{i.} = a\bar{y}_{..}$, we may rewrite Equation (2.6) as

$$RSSB = n\left(\sum_i \bar{y}_{i.}^2 - a\bar{y}_{..}^2\right).$$

We may take the expectation into the bracket and within the sum. Since both "estiamtors" are Gaussian distributed, we have $\langle \bar{y}_{i.}^2\rangle = \mu_i^2 + \sigma^2/n$[†] and $\langle \bar{y}_{..}^2\rangle = \mu^2 + \sigma^2/(n * a)$. We thus get

$$\begin{aligned} \langle RSSB\rangle &= n\left(\sum_i\left(\mu_i^2 + \frac{\sigma^2}{n}\right) - a\mu^2 - \frac{\sigma^2}{n}\right) \\ &= \sigma^2(a-1) + n\left(\sum_i \mu_i^2 - a\mu^2\right). \end{aligned}$$

The proof is completed by applying Lemma 2.1.1 once again. We reexpress $\sum_i \mu_i^2 - a\mu^2 = \sum_i(\mu_i - \mu)^2$ and thus arrive at the relation in Equation (2.10). □

2.2.2 Null hypothesis and test

The null hypothesis of the one way fixed effects ANOVA is that all α_i are zero. The following lemma helps to determine the appropriate significance test.

Lemma 2.2.2 (All MSE's estimate σ^2). *Under the null hypothesis $\alpha_i = 0\ \forall i$, all three mean square errors estimate the variance of the noise model, σ^2.*

[*]This is actually more a Bayesian than a frequentist perspective of what we do!

[†]Note that $\mu_i = \mu + \alpha_i$.

Proof for $MSEB$. This Lemma is obvious for $MSET$ and $MSEW$. The proof for the residual sum of squares between levels follows from Lemma 2.2.1. We first note that the expectation of $MSEB$ is

$$\langle MSEB \rangle = \frac{1}{dfB} \langle RSSB \rangle$$

However under the null hypothesis the expectation $\langle RSSB \rangle$ is just $\sigma^2(a-1)$ and since $dfB = a - 1$, we established the result. □

Lemma 2.2.2 allows to test the null hypothesis of the one way ANOVA with fixed effects by an F-test [5]. The ratio of mean square error between classes and within classes is distributed according to an F-distribution with $a-1$ and $a(n-1)$ degrees of freedom.

$$\frac{MSEB}{MSEW} \sim \mathcal{F}\left(a-1, a(n-1)\right) \tag{2.11}$$

We may reject the null hypothesis using an appropriate threshold for the corresponding p-value.

2.2.3 Inference over Levels

If the data allows to reject the null hypothesis of the overall ANOVA, we might be further interrested in looking for significant differences among the means of different levels.

Comparisons in general

The most general approach of such tests are the so called *contratsts*, which are constrained linear combinations of the different means.

$$L \;=\; \sum_i \lambda_i \bar{y}_{i.} \quad \text{with} \quad \sum_i \lambda_i = 0 \tag{2.12}$$

The standard error of the contrast is $se(L) = \sqrt{\sum_i \lambda_i^2}\sigma/\sqrt{n}$, with degrees of freedom dfW[‡]. Contrasts allow to compare groupings among classes. Assuming that there are 4 levels, we could for example test whether the average of the first two classes is significantly different from the average of the other two classes. This is done using $\lambda_1 = \lambda_2 = 0.5$ and $\lambda_3 = \lambda_4 = -0.5$, and testing whether zero is within $L \pm t_p se(L)$, where t_p denotes the desired p-quantile of the standard t-distribution with dfW degrees of freedom. Alternatively we can calculate the p-value of L as twice the value of the cumulative distribution function of a t-distribution with mean zero, standard deviation $se(L)$ and dfW degrees of freedom.

$$p_L = 2\mathrm{cdf}_{\mathcal{T}}(L; 0, se(L), dfW) \tag{2.13}$$

Pairwise comparisons are just a special case of contrasts, with $\lambda_i = 1$ and $\lambda_j = -1$ and all other $\lambda_k = 0$. In that case we get $L = \bar{y}_{i.} - \bar{y}_{j.}$, $se(L) = \sigma\sqrt{2/n}$ and thus the p-values of a pairwise t-test.

[‡]If we reject the null hypothesis of the ANOVA model, $MSEW$ estimates σ^2 and it does so with dfW degrees of freedom.

Error rates and multiple testing

In order to decide whether a difference is significant, we have to use a threshold on the p-value and report significance if p_L is below that threshold. This simple thresholding method is referred to as least significant difference or *LSD*. Thresholding inevitably results in wrong decisions. A mistake, where we erroneously report a significant difference is called Type I Error. If we use a 0.05 threshold and *one* test, we have exactly 5% chance of getting such an error.

Multiple testing obviously increases the chance of making *at least one* Type I Error. Assuming that two tests A and B are independent, the probability of making *not* a Type I Error is $P(A) = P(B) = 1 - \alpha'$, provided that we threshold with α'. Since the tests are independent the probability of making no Type I Error in test A *and* test B is therefore $P(A, B) = P(A)P(B) = (1 - \alpha')^2$. In the more general case of c comparisons with threshold α', we thus get the probability α of making *at least one* Type I Error (Family Wise Type I Error or FWE).

$$\begin{aligned} \alpha &= 1 - (1 - \alpha')^c \qquad (2.14) \\ &\approx c\alpha' \end{aligned}$$

A simple way of controling FWE is to multiply the individual p-values p_L from Equation (2.13) by the number of comparisons. This approach is referred to as Bonferroni correction and results in a very conservative test with a large Type II Error (i.e. errounoulsy not rejecting the null hypothesis). Alternative methods to reduce the overall Type I Error are the Scheffee test, Tukey's honest significant difference (HSD) or the false discovery rate.

2.3 One Way ANOVA with Random Effects

A random effects model must be used if the number of levels in the experiment is not exhaustive. A commonly used guideline is that one regards an effect as random if the number of levels is roughly around 10% or below of the number of levels in the population. The name captures the idea that the levels used in the experiment are a random draw from the population of all levels. In this case it makes no sense to regard the parameters of each level as a fixed quantity, since from this assumption no generalisation is to be gained (i.e. we do not learn anything about the population!). The only question one might ask is whether the effect contributes significantly to the overall variation and how much might be gained by having more observations per level.

$$y_{ij} = \mu + A_i + \epsilon_{ij} \qquad (2.15)$$

We assume that the paramaters of the random effects are random variables $A_i \sim \mathcal{N}(0, \sigma_A)$, and as before $\epsilon_{ij} \sim \mathcal{N}(0, \sigma)$. Parameter μ is a fixed parameter and - most important - we assume independence between ϵ_{ij} and A_i. The variance of y_{ij} is $var(y_{ij}) = \sigma_A^2 + \sigma^2$.

2.3.1 Residual sums of squares

The equations for the residual sums of squares, degrees of freedom and consequently the mean square errors of the random effects ANOVA are the same as those we used for the fixed effects ANOVA in section 2.2.1. However the residual sum of squares between classes, $RSSB$ estimates a different quantity.

Lemma 2.3.1 (Estimation by $RSSB$ in random effects ANOVA). *In a random effects ANOVA, the residual sum of squares $RSSB$ in Equation (2.6) estimates*

$$(a-1)(\sigma^2 + n\sigma_A^2) \tag{2.16}$$

Proof. First we note that $\bar{y}_{i.}$ and $\bar{y}_{..}$ are entirely obtained by summation and scaling and of independent Gaussian random variables. Following the rules of probability theory, we thus get

$$p(\bar{y}_{i.}) = \mathcal{N}\left(\mu, \sqrt{\frac{n\sigma_A^2 + \sigma^2}{n}}\right) \quad \text{and} \quad p(\bar{y}_{..}) = \mathcal{N}\left(\mu, \sqrt{\frac{n\sigma_A^2 + \sigma^2}{na}}\right)$$

and therefore

$$\left\langle \bar{y}_{i.}^2 \right\rangle = \mu^2 + \frac{n\sigma_A^2 + \sigma^2}{n} \quad \text{and} \quad \left\langle \bar{y}_{..}^2 \right\rangle = \mu^2 + \frac{n\sigma_A^2 + \sigma^2}{na}$$

Using this we obtain the expected residual sum of squares between classes as

$$\begin{aligned} \langle RSSB \rangle &= n\left(a\mu^2 + a\sigma_A^2 + a\sigma^2/n - a\mu^2 - \sigma_A^2 - \sigma^2/n\right) \\ &= \left(n\sigma_A^2 + \sigma^2\right)(a-1), \end{aligned}$$

which completes the proof. □

A trivial consequence of Lemma 2.3.1 is therefore that the mean square error between levels of the one way ANOVA with random effects, $MSEB$, estimates $n\sigma_A^2 + \sigma^2$.

2.3.2 Null Hypothesis and Test

The null hypothesis of the one way ANOVA with random effects is that the variance between classes, σ_A^2 is zero. In this case we get a similar relation as with the fixed effects model, since again all mean square errors estimate the variance of the error term σ^2. We can therefore use exactly the same test procedure as for the fixed effects model in Equation (2.11). Note though that the conclusion differs since we now test for zero variance. The random effects ANOVA does not lead to further inspection of the levels.

2.4 Mixed effects ANOVA

The mixed effects ANOVA (sometimes also called a mixed model ANOVA) is a multi way ANOVA, that is we have more than one classification. As an example we use a model with two fixed and two random effects. To put that in context with microarray analysis, we assume a classification in genes, time, biological smaple and technical replicate. For reasons that are made clear later, we use the model

$$y_{ijkl} = \mu + \alpha_i + \beta_{ij} + A_{ijk} + \epsilon_{ijkl}, \tag{2.17}$$

where μ models the global mean, α_i models the main gene effect, β_{ij} models the gene−time interaction, A_{ijk} models the gene−time−sample interaction and ϵ_{ijkl} models the residual error. The effects that correspond to gene and time are fixed. The effects corresponding to sample and replicate are random. Our discussion of the mixed effects ANOVA assumes a balanced design.

$$\begin{aligned} i &= 1,..,n & &- n\text{: number of genes} \\ j &= 1,..,m & &- m\text{: number of time points} \\ k &= 1,..,a & &- a\text{: number of samples} \\ l &= 1,..,b & &- b\text{: number of technical replicates} \end{aligned}$$

We also need the identifiability constraints $\sum_i \alpha_i = 0$ and $\forall i, \sum_j \beta_{ij} = 0$. We have $\epsilon_{ijkl} \sim \mathcal{N}(0,\sigma)$ and $A_{ijk} \sim \mathcal{N}(0,\sigma_A)$ and assume independence between ϵ_{ijkl} and A_{ijk}.

2.4.1 Residual Mean Square Error

We get the residual error

$$\begin{aligned} RSSE &= \sum_{ijkl} (y_{ijkl} - \bar{y}_{ijk.})^2 \\ &= \sum_{ijk} \left(\sum_l y_{ijkl}^2 - b\bar{y}_{ijk.}^2 \right) \end{aligned} \tag{2.18}$$

Observation y_{ijkl} is a sum of 3 constants μ, α_i and β_{ij} and two random variables A_{ijk} and ϵ_{ijkl}. The distribution is thus

$$p(y_{ijkl}) = \mathcal{N}\left(\mu + \alpha_i + \beta_{ij}, \sigma_A^2 + \sigma^2\right).$$

The sample mean $\bar{y}_{ijk.}$ is a sum over b i.i.d. random variables ϵ_{ijkl} and a scaling of μ, α_i, β_{ij} and A_{ijk} with b that is then scaled by $1/b$. The distribution is therefore

$$p(\bar{y}_{ijk.}) = \mathcal{N}\left(\mu + \alpha_i + \beta_{ij}, \frac{b\sigma_A^2 + \sigma^2}{b}\right).$$

Using the expectations $\langle y_{ijkj}^2 \rangle = (\mu + \alpha_i + \beta_{ij})^2 + \sigma_A^2 + \sigma^2$ and $\langle \bar{y}_{ijk.}^2 \rangle = (\mu + \alpha_i + \beta_{ij})^2 + \frac{b\sigma_A^2 + \sigma^2}{b}$ we get the expectation of $RSSE$ in Equation (2.18)

$$\langle RSSE \rangle = \sigma^2 nma(b-1) \ \text{ with } \ dfE = nma(b-1) \tag{2.19}$$

degrees of freedom. We immediately see that $\langle MSE \rangle$ estimates σ^2 (the residual variance).

2.4.2 Mean Square Between Levels of Effect A

We define

$$RSSA = b\sum_{ij}\left(\sum_{k}\bar{y}^2_{ijk.} - a\bar{y}^2_{ij..}\right). \tag{2.20}$$

Using the definition for $p(\bar{y}_{ijk.})$ from the previous section, we get

$$p(\bar{y}_{ij..}) = \mathcal{N}\left(\mu + \alpha_i + \beta_{ij}, \frac{b\sigma_A^2 + \sigma^2}{ab}\right)$$

by sumation over a i.i.d. $\bar{y}_{ijk.}$ random variables and scaling with $1/a$. This immediately gives the expecttaion $\left\langle\bar{y}^2_{ij..}\right\rangle = (\mu + \alpha_i + \beta_{ij})^2 + \frac{b\sigma_A^2+\sigma^2}{ab}$ and thus

$$\langle RSSA\rangle = nm(a-1)\left(b\sigma_A^2 + \sigma^2\right) \quad \text{with} \quad df A = nm(a-1) \tag{2.21}$$

The mean square MSA thus estimates $b\sigma_A^2 + \sigma^2$. As a note we should mention that the standard error

$$se(\bar{y}_{ij..}) = \frac{b\sigma_A^2 + \sigma^2}{ab}.$$

Hence MSA is directly relevant for testing contrasts of the gene-time interacton effect β_{ij}. Since $\langle MSA\rangle = b\sigma_A^2 + \sigma^2$, the null hypothesis that $\sigma_A = 0$ has the consequence that MSA estimates σ^2. We may thus test this hypothesis by an F-test

$$\frac{MSA}{MSE} \sim \mathcal{F}\left(nm(a-1), nma(b-1)\right).$$

2.4.3 Mean Sqare Between Levels of Effect β

We define the residual sum of squares between levels of effect β as

$$RSS\beta = ab\left(\sum_{ij}\bar{y}^2_{ij..} - m\bar{y}^2_{i...}\right), \tag{2.22}$$

where $\bar{y}_{i...}$ is obtained as sum over m i.i.d. random variavles $\bar{y}_{ij..}$ and scaling by $1/m$. The distribution $p(\bar{y}_{1...}) = \mathcal{N}(\mu+\alpha_i, \frac{b\sigma_A^2+\sigma^2}{mab})$ results in the expectation $\langle\bar{y}^2_{i...}\rangle = (\mu + \alpha_i)^2 + \frac{b\sigma_A^2+\sigma^2}{mab}$, which we use together with $\left\langle\bar{y}^2_{ij..}\right\rangle$ from the last section to obtain

$$\langle RSS\beta\rangle = n(m-1)\left(\sigma_A^2 + \sigma^2\right) + nmab\sum_{ij}\beta^2_{ij} \quad \text{with} \quad df\beta = n(m-1) \tag{2.23}$$

The mean square thus estimates

$$\langle MS\beta\rangle = b\sigma_A^2 + \sigma^2 + \frac{abm}{n(m-1)}\sum_{ij}\beta^2_{ij}$$

Hence under the null hypothesis that all β_{ij} are zero the $\langle MS\beta\rangle$ estimates $b\sigma_A^2 + \sigma^2$. We can thus testthis null hypothesis with the F-test

$$\frac{MS\beta}{MSA} \sim \mathcal{F}\left(n(m-1), nm(a-1)\right).$$

If we can reject this null hypothesis, further tests on the interaction levels are based on contrats among those and the standard error $se(\bar{y}_{ij..}) = \sqrt{\frac{b\sigma_A^2+\sigma^2}{ab}}$. All results of the previous discussion about multiple testing apply.

2.4.4 Mean Square Between Levels of α

We define

$$RSS\alpha = mab\left(\sum_i \bar{y}_{i\ldots}^2 - n\bar{y}_{\ldots.}^2\right) \tag{2.24}$$

The distribution of $\bar{y}_{\ldots.}$ is obtained by summation over n i.i.d. random variables $\bar{y}_{i\ldots.}$ and scaling with $1/m$ to give $p(\bar{y}_{\ldots.}) = \mathcal{N}\left(\mu, \frac{b\sigma_A^2+\sigma^2}{nmab}\right)$. Using $\langle\bar{y}_{\ldots.}^2\rangle = \mu^2 + \frac{b\sigma_A^2+\sigma^2}{nmab}$ and $\langle\bar{y}_{i\ldots}^2\rangle$ from the previous section, we get

$$\langle RSS\alpha\rangle = mab\left((n-1)\frac{b\sigma_A^2\sigma^2}{mab} + \sum_i \alpha_i^2\right) \quad \text{with} \quad df\alpha = n-1. \tag{2.25}$$

The mean square $MS\alpha$ thus estimates

$$\langle MS\alpha\rangle = \frac{mab}{n-1}\sum_i \alpha_i^2 + b\sigma_A^2 + \sigma^2.$$

In the case of the null hypothesis that all α_i are zero, this reduces to $b\sigma_A^2 + \sigma^2$. We can thus again test the hypothesis against an F-statistic:

$$\frac{MS\alpha}{MSA} = \mathcal{F}\left(n-1, nm(a-1)\right).$$

If we reject the null hypothesis further pairwise comparisons might be carried out to find those levels that differ significantly from a reference level. This involves contrasts and the standard error $se(\bar{y}_{i\ldots}) = \sqrt{\frac{b\sigma_A^2+\sigma^2}{mab}}$.

Chapter 3

FSPMA Tutorial

3.1 Introduction

This tutorial was generated from "fspma.Rnw" using Sweave [1]. All experiments can thus be run as discussed below, once the library has been installed and the relevant archives have been copied to a local directory. We illustrate five different scenarios that can emerge in microarray analysis. We analyse experiments with data from two colour arrays (CDNA or two colour oligo arrays), from one channel arrays (e.g. as one obtains from the Affymetrix platform) or with data that underwent some form of preprocessing. Here we provide data that is already on a transformed scale and another experiment on data that was normalised before. Finally we provide examples that illustrate FSPMA's data visualisation capabilities and an interaction of FSPMA based microarray analysis and the use of functions from other libraries. The following table helps establishing the relation between the archives in the "$FSPMAHOME$/doc/" directory and the examples it represents. The symbol "$FSPMAHOME$" refers to the FSPMA installation directory. It can be found easily using R's online help and following the link "Packages" and "fspma" and "browse directory". R's online help is found in the R_HOME/doc/html directory (there one opens index.html under Linux or rwin.html under Windows). Windows users are though advised to start with R's help menue in the graphical user interface.

Experiments	archive
two channel analysis and FSPMA based visualisations	"twochannel.zip"
single channel analysis	"singlechannel.zip"
analysis of transformed data (e.g. log or variance stabilising)	"logsingle.zip"
analysis of normalised data	"normsingle.zip"
interfacing with other libraries	"libinterface.zip"

On machines that have a working "make" command available, the "Makefile" in the "$FSPMAHOME$/doc/" directory allows to automate running all examples in the Sweave file "fspmatutorial.Rnw". After copying all zip archives and the make file into a local directory, the commands "make preprnwrun" and "make fspmatutorial.pdf" will run Sweave and convert the resulting LaTex source into a pdf file. Note that running Sweave like that (via an R batch call) will take about 5 minutes and not produce any visual output. The data in all examples is meant for illustration purpouse. It consists of a short subset of genes that were measured in a mouse testis time course. All datasets have been constructed from the same origin and thus give identical results. This excludes the last example where we show how to merge FSPMA with other libraries using some functions of YASMA [4] as an example.

3.2 Remark on Computational Complexity

Before we start describing these experiments we would like to add a note on the computational complexity of FSPMA's algorithms. In this tutorial data is only a very small subset of what we would expect in real experiments and there are no timing problems. In general things are not too time consuming, however there are two exceptions. If one specifies a "base level comparison", this acts as a short cut for all pair wise contrasts that involve this level. The algorithm thus calculates p-values of a matrix of number of genes times number of levels in the rank effect minus one, which takes some time. The other time consuming operation is imputing by k nearest neighbours. This can, depending on the number of samples that are imputed, take up to a couple of hours for a "real world" dataset.

3.3 Analysis of two channel data

The first experiment illustrates how to use FSPMA's definition file for analysing two channel data from a *balanced* time course experiment. The requirement of a balanced experimental design results form the fact that we internally use YASMA. Compared to other packages YASMA has the advantage of allowing a precise specification of nested effects. We obtain therefore more realistic p-values. The following code fragment will load the data and perform all analysis steps. The example generates a set of output files that rank genes that were after p-value adjustment found to be significant under the specified threshold.

```
> library(fspma)

Loading required package: yasma
Loading required package: nlme

> ret <- fspma.wrapper("twochannel.def")

Run started on - Mon Jun 12 2006 11:48:12 AM

Loading info from file.
Checking consistency of the definition file twochannel.def was successful.
Loading RG data.
Reading File: R35_NIA1_AWX_ad_612_Fl_output.tsv
Reading File: R35_NIA1_AWX_ad_613_Fl_output.tsv
Reading File: R35_NIA1_AWX_ad_629b_Fl_output.tsv
Reading File: R35_NIA1_AWX_ad_630_Fl_output.tsv
Reading File: R25_NIA1_AWX_d01_s16_Fl_output.tsv
Reading File: R25_NIA1_AWX_d01_s17_Fl_output.tsv
Reading File: R25_NIA1_AWX_d01_s18_Fl_output.tsv
Reading File: R25_NIA1_AWX_d01_s21_Fl_output.tsv
Reading File: R25_NIA1_AWX_d05_s13_Fl_output.tsv
Reading File: R35_NIA1_AWX_d05_545_Fl_output.tsv
Reading File: R35_NIA1_AWX_d05_546_Fl_output.tsv
Reading File: R35_NIA1_AWX_d05_547_Fl_output.tsv
Reading File: R25_NIA1_AWX_d10_s28_Fl_output.tsv
Reading File: R35_NIA1_AWX_d10_596_Fl_output.tsv
Reading File: R35_NIA1_AWX_d10_597_Fl_output.tsv
Reading File: R35_NIA1_AWX_d10_600_Fl_output.tsv
Reading File: R35_NIA1_AWX_d15_594_Fl_output.tsv
Reading File: R35_NIA1_AWX_d15_605_Fl_output.tsv
Reading File: R35_NIA1_AWX_d15_671_Fl_output.tsv
Reading File: R35_NIA1_AWX_d15_674_Fl_output.tsv
Reading File: R35_NIA1_AWX_d23_653_Fl_output.tsv
Reading File: R35_NIA1_AWX_d23_654_Fl_output.tsv
Reading File: R35_NIA1_AWX_d23_655b_Fl_output.tsv
Reading File: R35_NIA1_AWX_d23_670_Fl_output.tsv
Reading File: R35_NIA1_AWX_d35_631_Fl_output.tsv
Reading File: R35_NIA1_AWX_d35_632_Fl_output.tsv
Reading File: R35_NIA1_AWX_d35_633_Fl_output.tsv
Reading File: R35_NIA1_AWX_d35_651_Fl_output.tsv
Number of NA entries from channel:   0.
Number of NA entries from flag:   0 (NA spikes):   0
Number of NA entries after imputing:   0
Getting model.info
Do classical normalization.
Normalising by: location
Normalising by: scale
Write raw data to files.
Convert RG data to log ratios.
Calculating ANOVA and write table and variance components.
Loading required package: MASS
...Getting number of comparisons
Ranking by Variety.
ANOVA ranking.
Contrast based ranking
base:   1 - shortcut for several contrasts, be patient!
contrast: -0.333333333333333,0.25,0.25,0.25,0.25,-0.333333333333333,-0.3333333333
Processing completed.
```

The entire logic of microarray analysis with FSPMA is thus packed into the definition file and no coding is required to adapt analysis to different types of experiments, of course within the limits of the underlying package, which restricts it to balanced reference designs.

3.3.1 Log file of the fspma.wrapper run

Calling **fspma.wrapper** as shown in the previous R code fragment will produce several files as output. One of these files is an ASCII text file that stores log information. This log file is intended to provide a protocol of all steps taken during analysis. It contains in essence the same output, one obtains in the R console window. The default filename of this log file is "fspma.log.txt". It may be changed by overriding the **log.fname** parameter of **fspma.wrapper**. The log-file is in particular important to debug definition files. After parsing the file, **fspma.wrapper** checks the information for consistency. All inconsistencies are reported and written into the log file. Most problems should therefore be caught before the microarray data gets loaded, which is important, because up following analysis can take some time.

3.3.2 Comments on "twochannel.def"

The experiment

The first part of any analysis is to specify the experimental layout. Here we have a time course experiment with 7 time points, 4 biological replicates and 2 on slide replicates per gene. If some of the genes have more replicates, one specifies the minimum number and FSPMA's data loader discards the excess in essence randomly. Otherwise the result would be biased.

```
# Number of levels of all effects that appear in the experiment. The last
# is the minimum number of replicates per slide and must be specified.
Effects:        7       4       2
# Is experiment Unbalanced:
Is.Unbalanced:  F
# Effect Names: We have thus 5 time points and 2 samples per time point
Eff.Nams:       time    sample  rep
# of which sample and replicate on slide are random effects (all that are not listed exhaustively)
RandEffs:       F       T       T
# over which effect should be ranked - We rank over time here.
Rank.Eff:       1
# Names of each ranking dimension (will become R variable names)
Rank.Names:     adult   day.1   day.5   day.10  day.15  day.23  day.35
```

It is vital to understand the difference between random and fixed effects: from the experimental point of view random effects are *not listed exhaustively.* For random effects, statements about significant differences of averages make no sense, since there are many other levels that did not find their way into the analysis. Fixed effects on the other hand are such, where all levels are listed exhaustively (e.g. dye swap), or levels not covered by the analysis are not of interest. The rank effect - in this case time - is always a fixed effect. Hence here, we do not aim at conclusions that generalise to time points *not* covered by the experiment. Note that we have 7 time points, all have a “rank name”. This is a name that should allow an easy identification of a level. Depending on how we analyse the data, rank names are possibly used in output file names, to refer to a particular pair wise comparison and as column headers in these files, to identify the average log ratios found at that level. The assignment of measurements (i.e. slide files) to every combination of levels is discussed later.

Contrasts and output files

The most important result of every FSPMA run is a set of (tab delimited) ASCII text files. We get two files that store normalized data and effects in a row format. Each row corresponds to a known effect combination and all corresponding expression values. Gene names are identified by the column header. The main reason for providing this output is to convert data to a standardised format that can be used as a starting point for other types of data analysis. Processing “twochannels.def”, we get this information in *twochannel_data_normlogrt.tsv* and *twochannel_data_normeffdesc.tsv*. The next output concerns the ANOVA table and the variance components. They are here written into a text file named *twochannel_aov.txt*. Finally each comparison generates a rank list of significant genes as tab delimited text file. The p-values are adjusted for multiple comparisons, where **fspma.wrapper** uses the number of all comparison that were specified in one definition file for this adjustment. It is thus important to place all comparisons in one file - otherwise the adjustments are not correct. This analysis generates *anova.rank.twochannel_comp.tsv* for the ANOVA based ranking, *test.variety.twochannel_comp.tsv* for the ranking based on average channel log ratios, and *test.day.1.bladulttwochannel_comp.tsv* to *test.day.35.bladulttwochannel_comp.tsv* for all pair wise comparisons of the adult generation against all other time points. We also provide a more general comparison that is based on a contrast. This comparison tests for significant differences when comparing the average log ratios of day 23, day 35 and adult against those of day 1 up to day 15. The result of this test is stored in file *test.ct.ad23.d1d15.twochannel_comp.tsv*. The filenames are specified by definition file entries, which in case of all rank tables are generated by concatenating the contrast name (or a string derived thereof) and the corresponding base filename entry (Contrast.Outfile:) in the definition file. FSPMA uses “anova.rank” for anova based ranking and “test.variety” for the test of average channel differences. Base level comparisons are short cuts for all binary comparisons of the base level against the remaining levels. In this case FSPMA concatenates the base level name (bladult) and the corresponding rank name (e.g. test.day.1.bladult). If we want to test for significant differences w.r.t. states that are known to be constant for more than one time point, we have to use a more general contrast. As an example, we might be interested to test for significant differences in the average log ratios of day35 and adult agaist the average log ratios of all other time points. Such comparisons can be specified as vector of 0’s, -1’s and 1’s. FSPMA requires such contrasts to be named (the name is used to generate the output filename) and to specify a vector of as many 0’s, 1’s and -1’s, as there are levels in the rank effect. Zero entries are excluded from the comparison. All ones and minus ones are grouped, to form a statistic that is used to obtain p-values of the null hypothesis that there is no significant difference between the groups. The result of this comparison is written into file *test.ct.ad23.d1d15.twochannel_comp.tsv*. The relevant definition file entries are:

```
# We may also specify a comparison of average difference in d35 and adult versus all earlier
# development stages. (This is useful to rank w.r.t. biological states that are known to be constant
# for certain development periods.)
Base.Contrast:   ct.ad23.d1d15    -1     1     1     1     1     -1     -1
# Example for general ranking against first time point
# bladult : base level adult (here a short cut for 6 pair wise contrasts − > expect some
# computation if your data contains the full transcriptome!).
Base.Contrast:   bladult  1
# Anova based ranking : for each gene we calculate an anova and test
# whether all means are identical.
Base.Contrast:   ANOVA
# finally if we are interested in significant differences between colours we may chose VARIETY.
# This type of comparison corresponds to the yasma non bootsrap tests.
Base.Contrast:   VARIETY
# Output file to write the gene list from the comparison,
# use NA if not required extension .tsv is for tab separated
# files that load conveniently into MS Excel.
Contrast.Outfile:  twochannel_comp.tsv
# ANOVA table and variance component output file
ANOVA.Outfile:  twochannel_aov.txt
# The following entry allows to write the normalized log ratios to an output file.
# NA is for none. The string is actually only the base file name with various
# endings that accommodate the various bits of information.
DataOutFnam:   twochannel_data_norm
```

Allocating files to levels

The next step in preparing analysis is to allocate an input file to every combination of levels. This excludes on slide replicates, which are found and allocated automatically when the data is loaded. The relevant section of the definition file is found under "file.alloc:". Each line starts with a file name, lists the exhaustive combination of levels the file is allocated to and finally indicates, whether the slide is a dye swap. Note that this information is always necessary, even for single channel experiments, since this single column can as well be a log ratio information with dye swaps. The flag "load.fast:" allows to select fast data loading. If this flag is set "TRUE", FSPMA assumes that all gene positions that were found in the first file are valid in all subsequent files. Searching for genes is quite costly. Since seraching takes place only once, data loading is much more efficient. It is though the users responsibility to ensure that the gene positions are identical since any differences will lead to meaningless results. If there is any doubt, it is better to set this flag “FALSE” and give the analysis a little more time.

```
# Do a fast file load? If one sets load.fast: to TRUE, fspma assumes that
# the positions of genes found in the first file is identical with the
# gene positions in all subsequent files. This speeds up data loading considerably.
load.fast: F
file.alloc:
# reminder - the last effect is always replicate on slides and is thus *not* allocated
# to a particular file. (Replicates are assigned by data loading).
# file name:                             time   sample   dye swap?
# adult generation
R35_NIA1_AWX_ad_612_Fl_output.tsv        1      1        F
R35_NIA1_AWX_ad_613_Fl_output.tsv        1      2        F
R35_NIA1_AWX_ad_629b_Fl_output.tsv       1      3        F
R35_NIA1_AWX_ad_630_Fl_output.tsv        1      4        F
# day one
R25_NIA1_AWX_d01_s16_Fl_output.tsv       2      1        F
R25_NIA1_AWX_d01_s17_Fl_output.tsv       2      2        F
R25_NIA1_AWX_d01_s18_Fl_output.tsv       2      3        F
R25_NIA1_AWX_d01_s21_Fl_output.tsv       2      4        F
# day 5
R25_NIA1_AWX_d05_s13_Fl_output.tsv       3      1        F
R35_NIA1_AWX_d05_545_Fl_output.tsv       3      2        F
R35_NIA1_AWX_d05_546_Fl_output.tsv       3      3        F
R35_NIA1_AWX_d05_547_Fl_output.tsv       3      4        F
# day 10
R25_NIA1_AWX_d10_s28_Fl_output.tsv       4      1        F
R35_NIA1_AWX_d10_596_Fl_output.tsv       4      2        F
R35_NIA1_AWX_d10_597_Fl_output.tsv       4      3        F
R35_NIA1_AWX_d10_600_Fl_output.tsv       4      4        F
# day 15
R35_NIA1_AWX_d15_594_Fl_output.tsv       5      1        F
R35_NIA1_AWX_d15_605_Fl_output.tsv       5      2        F
R35_NIA1_AWX_d15_671_Fl_output.tsv       5      3        F
R35_NIA1_AWX_d15_674_Fl_output.tsv       5      4        F
# day 23
R35_NIA1_AWX_d23_653_Fl_output.tsv       6      1        F
R35_NIA1_AWX_d23_654_Fl_output.tsv       6      2        F
R35_NIA1_AWX_d23_655b_Fl_output.tsv      6      3        F
R35_NIA1_AWX_d23_670_Fl_output.tsv       6      4        F
# day 35
R35_NIA1_AWX_d35_631_Fl_output.tsv       7      1        F
R35_NIA1_AWX_d35_632_Fl_output.tsv       7      2        F
R35_NIA1_AWX_d35_633_Fl_output.tsv       7      3        F
R35_NIA1_AWX_d35_651_Fl_output.tsv       7      4        F
```

Assigning RG columns

The next step in the experiment description is to assign column headers of the input file to the corresponding RG columns. The minimal setting for two channel arrays is to assign the gene id column, the R and the G channel. As is illustrated below for the R-background signal, unused columns are identified by using "NA" as specification.

```
# gene column name in the header
gene.colnam:        NAME
# Cy5 spot column name in the header
R.colnam:           AMPCH1
# Cy5 background column name in the header (NA if none)
Rb.colnam:          NA
# Cy3 spot column name in the header (NA if none) - for single channel experiments
G.colnam:           AMPCH2
```

Gene list

To allow exclusion of non genomic information like sub grid markers and to cope with situations where different genes appear on slides in different numbers, one has to provide an exhaustive list of all gene identifiers that should go into analysis.

```
# Finally a list of all gene names (as found in gene.colnam in the data)
# which should go into the experiment. This is necessary to allow for
# experiment designs where people use different numbers of replicates
# on slide and to get rid of all unwanted entries like marker spots etc.
genes.list:
H3078A06
H3078C06
H3078E06
H3078G06
H3078A12
H3078C12
H3078E12
H3078G12
Ctl141002_01_A12
Ctl141002_01_E12
Ctl141002_01_I12
Ctl141002_01_M12
Ctl141002_01_A24
Ctl141002_01_E24
Ctl141002_01_I24
Ctl141002_01_M24
```

H3066C12
H3066E1
... and more

Normalisation

Normalisation is here done with classical methods. Since we also want to allow log transformed data, we use in this situation a slightly modified functionality of the underlying YASMA package. Note that FSPMA also allows spike based normalisation. Since we are at present not in the position to provide any data that have spikes on the array, we refer to explanations in [2] and to the online help for more information.

```
# normalization control NA if none, otherwise loess,
# location or scale combinations like loess scale or
# location scale are allowed as well. For loess scale the order matters!
Normalization:    location  scale
```

3.4 Single channel and preprocessed data

3.4.1 Single channel analysis

Analysis of single channel data follows exactly the same lines as were described above. The example provided here was generated from the same two channel data. At the R command line, we just type:

```
> library(fspma)
> ret <- fspma.wrapper("onechannel_affytype.def")

Run started on - Mon Jun 12 2006 11:48:20 AM

Loading info from file.
Checking consistency of the definition file onechannel_affytype.def was successfu
Loading RG data.
Reading File: sig_t1_s1.txt
Reading File: sig_t1_s2.txt
Reading File: sig_t1_s3.txt
Reading File: sig_t1_s4.txt
Reading File: sig_t2_s1.txt
Reading File: sig_t2_s2.txt
Reading File: sig_t2_s3.txt
Reading File: sig_t2_s4.txt
Reading File: sig_t3_s1.txt
Reading File: sig_t3_s2.txt
Reading File: sig_t3_s3.txt
Reading File: sig_t3_s4.txt
Reading File: sig_t4_s1.txt
Reading File: sig_t4_s2.txt
Reading File: sig_t4_s3.txt
Reading File: sig_t4_s4.txt
Reading File: sig_t5_s1.txt
Reading File: sig_t5_s2.txt
Reading File: sig_t5_s3.txt
Reading File: sig_t5_s4.txt
Reading File: sig_t6_s1.txt
Reading File: sig_t6_s2.txt
Reading File: sig_t6_s3.txt
Reading File: sig_t6_s4.txt
Reading File: sig_t7_s1.txt
Reading File: sig_t7_s2.txt
Reading File: sig_t7_s3.txt
Reading File: sig_t7_s4.txt
Number of NA entries from channel:   0.
Number of NA entries from flag:   0 (NA spikes):   0
Number of NA entries after imputing:   0
Getting model.info
Do classical normalization.
Normalising by: location
Normalising by: scale
Write raw data to files.
Convert RG data to log ratios.
Calculating ANOVA and write table and variance components.
...Getting number of comparisons
Ranking by Variety.
ANOVA ranking.
Contrast based ranking
base:   1 - shortcut for several contrasts, be patient!
contrast: -0.333333333333333,0.25,0.25,0.25,0.25,-0.333333333333333,-0.33333333333333
Processing completed.
```

Apart from having used different output file names, to allow that the analysis results do not interfere with the other experiments, we have to adjust the channel name specification, where we chose the naming convention normally found in Affymetrix data. This definition file thus differs from the previous one, in that we have different channel names and file names, we allocate to the level combinations.

```
# gene column name in the header
gene.colnam:        Probe Set Name
# Cy5 spot column name in the header
R.colnam:           Signal
```

```
# Cy5 background column name in the header (NA if none)
Rb.colnam:        NA
# Cy3 spot column name in the header (NA if none) - for single channel experiments
G.colnam:         NA
# reminder - the last effect is always replicate on slides and is thus *not* allocated
# to a particular file. (Replicates are assigned by data loading).
# file name:                                        time    sample   dye swap?
# adult generation
sig_t1_s1.txt                                       1       1        F
sig_t1_s2.txt                                       1       2        F
sig_t1_s3.txt                                       1       3        F
sig_t1_s4.txt                                       1       4        F
# day one
sig_t2_s1.txt                                       2       1        F
sig_t2_s2.txt                                       2       2        F
sig_t2_s3.txt                                       2       3        F
sig_t2_s4.txt                                       2       4        F
# day 5
sig_t3_s1.txt                                       3       1        F
sig_t3_s2.txt                                       3       2        F
sig_t3_s3.txt                                       3       3        F
sig_t3_s4.txt                                       3       4        F
# day 10
sig_t4_s1.txt                                       4       1        F
sig_t4_s2.txt                                       4       2        F
sig_t4_s3.txt                                       4       3        F
sig_t4_s4.txt                                       4       4        F
# day 15
sig_t5_s1.txt                                       5       1        F
sig_t5_s2.txt                                       5       2        F
sig_t5_s3.txt                                       5       3        F
sig_t5_s4.txt                                       5       4        F
# day 23
sig_t6_s1.txt                                       6       1        F
sig_t6_s2.txt                                       6       2        F
sig_t6_s3.txt                                       6       3        F
sig_t6_s4.txt                                       6       4        F
# day 35
sig_t7_s1.txt                                       7       1        F
sig_t7_s2.txt                                       7       2        F
sig_t7_s3.txt                                       7       3        F
sig_t7_s4.txt                                       7       4        F
```

As the name suggests, this definition file is similar to a definition file we would specify for an Affymetrix type analysis. The main difference is that the number of replicates in Affy-data is 1, whereas in this artificially generated single channel data, we have 2.

3.4.2 Preprocessed data

The first example we provide here assumes that data is available as single channel log ratios (or log expressions, which is indifferent). We do though not assume that this data is normalized. FSPMA again just requires to specify an appropriately set up definition file.

```
> library(fspma)
> ret <- fspma.wrapper("onechannel_logdata.def")

Run started on - Mon Jun 12 2006 11:48:27 AM

Loading info from file.
Checking consistency of the definition file onechannel_logdata.def was successful.
Loading RG data.
Reading File: siglg_t1_s1.txt
Reading File: siglg_t1_s2.txt
Reading File: siglg_t1_s3.txt
Reading File: siglg_t1_s4.txt
Reading File: siglg_t2_s1.txt
Reading File: siglg_t2_s2.txt
Reading File: siglg_t2_s3.txt
Reading File: siglg_t2_s4.txt
Reading File: siglg_t3_s1.txt
Reading File: siglg_t3_s2.txt
Reading File: siglg_t3_s3.txt
Reading File: siglg_t3_s4.txt
Reading File: siglg_t4_s1.txt
Reading File: siglg_t4_s2.txt
Reading File: siglg_t4_s3.txt
Reading File: siglg_t4_s4.txt
Reading File: siglg_t5_s1.txt
Reading File: siglg_t5_s2.txt
Reading File: siglg_t5_s3.txt
Reading File: siglg_t5_s4.txt
Reading File: siglg_t6_s1.txt
Reading File: siglg_t6_s2.txt
Reading File: siglg_t6_s3.txt
Reading File: siglg_t6_s4.txt
Reading File: siglg_t7_s1.txt
Reading File: siglg_t7_s2.txt
Reading File: siglg_t7_s3.txt
Reading File: siglg_t7_s4.txt
Number of NA entries from channel:   0.
Number of NA entries from flag:   0 (NA spikes):   0
Number of NA entries after imputing:   0
Getting model.info
Do classical normalization.
Normalising by: location
Normalising by: scale
Write raw data to files.
Convert RG *log data* to log ratios (no log taken).
Calculating ANOVA and write table and variance components.
...Getting number of comparisons
Ranking by Variety.
ANOVA ranking.
Contrast based ranking
base:   1 - shortcut for several contrasts, be patient!
Processing completed.
```

The main difference to the previous example is that we need to indicate that expression values are already on a log scale. We thus do not move the data to the log scale.

```
# control conversion RG -> array (T -> take log, F -> convert as is)
# this data is on log scale (however not normalized)
DoRG2logarray:  F
```

Our next analysis of preprocessed data assumes that the data are already on a normalized log ratio scale. We thus do not move the data onto log ratio scale and specify that normalisation is not intended. The analysis is again done by using an appropriately set up definition file.

```
> library(fspma)
> ret <- fspma.wrapper("onechannel_lognorm.def")
```

```
Run started on - Mon Jun 12 2006 11:48:34 AM

Loading info from file.
Checking consistency of the definition file onechannel_lognorm.def was successful
Loading RG data.
Reading File: nrmlg_t1_s1.txt
Reading File: nrmlg_t1_s2.txt
Reading File: nrmlg_t1_s3.txt
Reading File: nrmlg_t1_s4.txt
Reading File: nrmlg_t2_s1.txt
Reading File: nrmlg_t2_s2.txt
Reading File: nrmlg_t2_s3.txt
Reading File: nrmlg_t2_s4.txt
Reading File: nrmlg_t3_s1.txt
Reading File: nrmlg_t3_s2.txt
Reading File: nrmlg_t3_s3.txt
Reading File: nrmlg_t3_s4.txt
Reading File: nrmlg_t4_s1.txt
Reading File: nrmlg_t4_s2.txt
Reading File: nrmlg_t4_s3.txt
Reading File: nrmlg_t4_s4.txt
Reading File: nrmlg_t5_s1.txt
Reading File: nrmlg_t5_s2.txt
Reading File: nrmlg_t5_s3.txt
Reading File: nrmlg_t5_s4.txt
Reading File: nrmlg_t6_s1.txt
Reading File: nrmlg_t6_s2.txt
Reading File: nrmlg_t6_s3.txt
Reading File: nrmlg_t6_s4.txt
Reading File: nrmlg_t7_s1.txt
Reading File: nrmlg_t7_s2.txt
Reading File: nrmlg_t7_s3.txt
Reading File: nrmlg_t7_s4.txt
Number of NA entries from channel:   0.
Number of NA entries from flag:   0 (NA spikes):   0
Number of NA entries after imputing:   0
Getting model.info
No normalization (data taken as is).
Write raw data to files.
Convert RG *log data* to log ratios (no log taken).
Calculating ANOVA and write table and variance components.
...Getting number of comparisons
Ranking by Variety.
ANOVA ranking.
Contrast based ranking
base:   1 - shortcut for several contrasts, be patient!
contrast: -0.333333333333333,0.25,0.25,0.25,0.25,-0.333333333333333,-0.333333333333333
Processing completed.
```

This definition file again uses different input and output file names, to allow that the analysis results do not interfere with other experiments. Since here normalisation was done before the data gets handed over to FSPMA, we do not normalise the data here. The main difference to the previous example is thus:

```
# here: no normalization since that was done before.
Normalization:     NA
```

3.5 Additional functionality of FSPMA

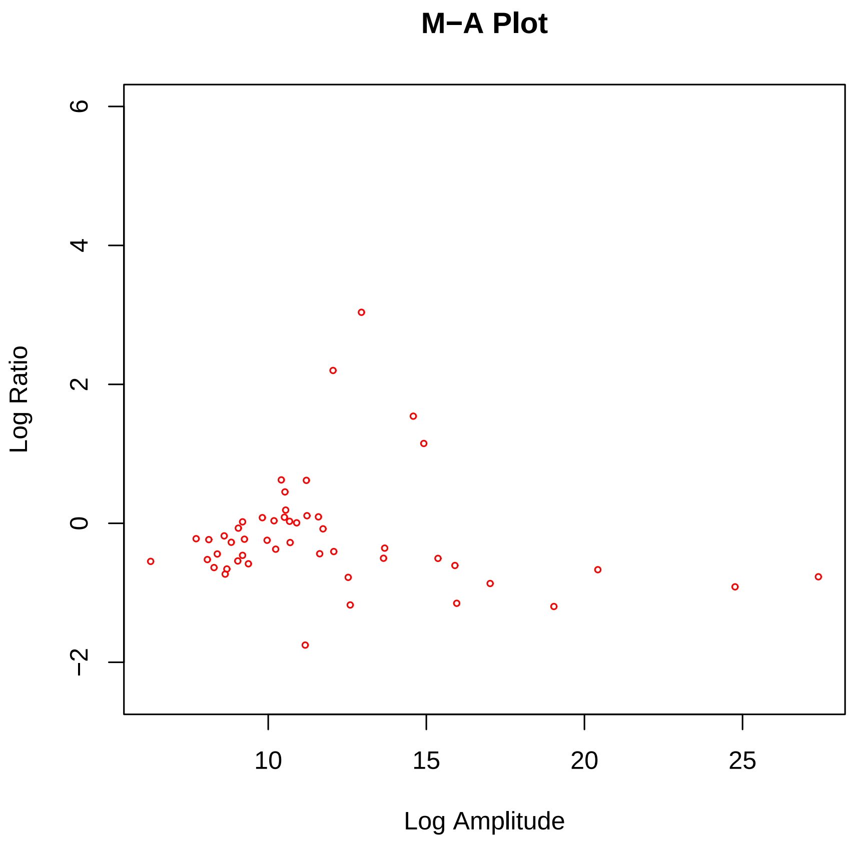
The main reasoning behind FSPMA is to provide an easy to use interface to loading data, normalization, data cleaning (i.e. remove or impute bad quality flagged expression values) and inference. There is however an additional aspect, one should consider during analysis. It is strictly recommended, to look at the data before relying on the result. For that purpouse FSPMA provides three different graphical views. Before we explore these, we need to generate a FSPMA object.

```
> library(fspma)
> ret <- fspma.wrapper("twochannel.def")

Run started on - Mon Jun 12 2006 11:48:41 AM

Loading info from file.
Checking consistency of the definition file twochannel.def was successful.
Loading RG data.
Reading File: R35_NIA1_AWX_ad_612_Fl_output.tsv
Reading File: R35_NIA1_AWX_ad_613_Fl_output.tsv
Reading File: R35_NIA1_AWX_ad_629b_Fl_output.tsv
Reading File: R35_NIA1_AWX_ad_630_Fl_output.tsv
Reading File: R25_NIA1_AWX_d01_s16_Fl_output.tsv
Reading File: R25_NIA1_AWX_d01_s17_Fl_output.tsv
Reading File: R25_NIA1_AWX_d01_s18_Fl_output.tsv
Reading File: R25_NIA1_AWX_d01_s21_Fl_output.tsv
Reading File: R25_NIA1_AWX_d05_s13_Fl_output.tsv
Reading File: R35_NIA1_AWX_d05_545_Fl_output.tsv
Reading File: R35_NIA1_AWX_d05_546_Fl_output.tsv
Reading File: R35_NIA1_AWX_d05_547_Fl_output.tsv
Reading File: R25_NIA1_AWX_d10_s28_Fl_output.tsv
Reading File: R35_NIA1_AWX_d10_596_Fl_output.tsv
Reading File: R35_NIA1_AWX_d10_597_Fl_output.tsv
Reading File: R35_NIA1_AWX_d10_600_Fl_output.tsv
Reading File: R35_NIA1_AWX_d15_594_Fl_output.tsv
Reading File: R35_NIA1_AWX_d15_605_Fl_output.tsv
Reading File: R35_NIA1_AWX_d15_671_Fl_output.tsv
Reading File: R35_NIA1_AWX_d15_674_Fl_output.tsv
Reading File: R35_NIA1_AWX_d23_653_Fl_output.tsv
Reading File: R35_NIA1_AWX_d23_654_Fl_output.tsv
Reading File: R35_NIA1_AWX_d23_655b_Fl_output.tsv
Reading File: R35_NIA1_AWX_d23_670_Fl_output.tsv
Reading File: R35_NIA1_AWX_d35_631_Fl_output.tsv
Reading File: R35_NIA1_AWX_d35_632_Fl_output.tsv
Reading File: R35_NIA1_AWX_d35_633_Fl_output.tsv
Reading File: R35_NIA1_AWX_d35_651_Fl_output.tsv
Number of NA entries from channel:   0.
Number of NA entries from flag:   0 (NA spikes):   0
Number of NA entries after imputing:   0
Getting model.info
Do classical normalization.
Normalising by: location
Normalising by: scale
Write raw data to files.
Convert RG data to log ratios.
Calculating ANOVA and write table and variance components.
...Getting number of comparisons
Ranking by Variety.
ANOVA ranking.
Contrast based ranking
base:   1 - shortcut for several contrasts, be patient!
contrast: -0.333333333333333,0.25,0.25,0.25,0.25,-0.333333333333333,-0.33333333333333
Processing completed.
```

3.5.1 Log ratio over log amplitude plots (M/A) plots

We can now use the FSPMA object to produce a scatter plot in log amplitude log ratio space of a randomly selected subset of genes on any one slide. This "M/A" representation is only useful for two channel input (since for single channel data M and A are identical).

```
> fspma.maplot(ret, slideno = 1)
```



Function **fspma.maplot** can be configured further. We may include spike genes if such are available or modify the plot title. Specifying a filename allows to write the output to an "eps" file (encapsulated postscript). These files can be used to illustrate documents. Details about other options in **fspma.maplot** can be found in the online help of the library.

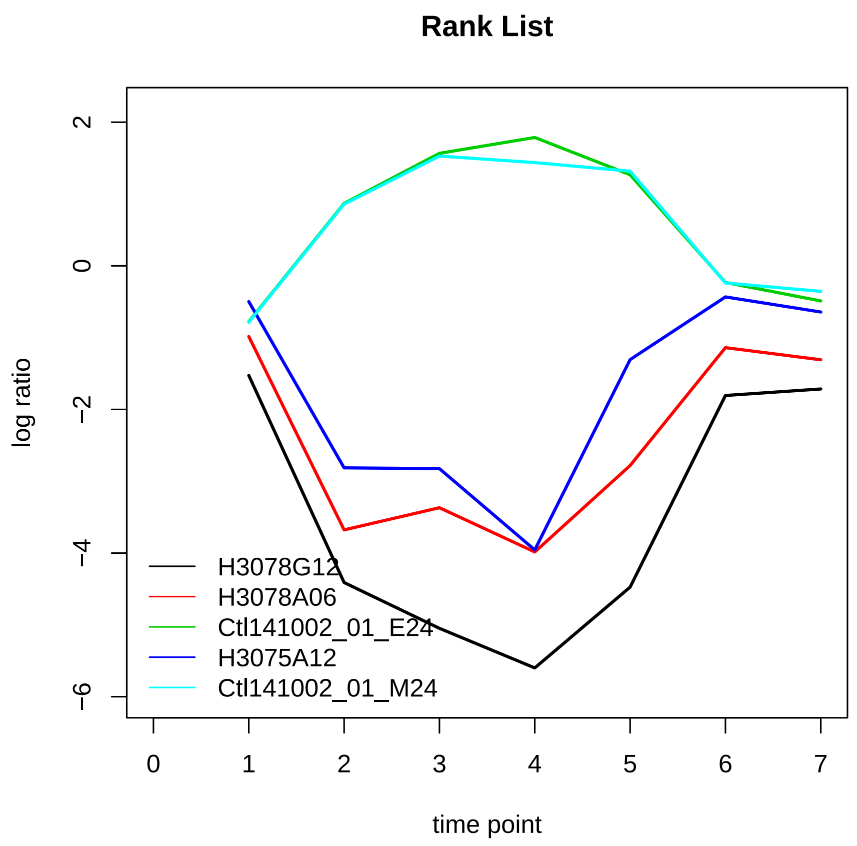
3.5.2 Plotting average log ratios of top ranked genes

To inspect the average log ratios (or log expressions), we may use **fspma.rankplot** to obtain a plot of log ratios over rank effects for the n-top ranked genes. This representation can be used for all data sources. We need to provide two parameters: a FSPMA object as returned by **fspma.wrapper** and the name of the list of interest. The names are identical to the first part of the rank file names constructed by the library. They can at any time be obtained by inspecting the names of the various rank lists. Sometimes it will be necessary to adjust the position of the legend within a plot. We move it here to the bottom left corner to avoid overlap with the graph.

```
> listnams <- names(ret$plt.tabs)
> cat(listnams[c(1:4)], sep = ", ")
```

```
test.day.1.bladult, test.day.5.bladult, test.day.10.bladult, test.day.15.bladult
```

```
> fspma.rankplot(ret, listnams[2], leg.pos = "bottomleft")
```



Again there are more options available. Details are found in the online help.

3.5.3 Scatter plots of average channel intensities and log ratio differences

To allow these scatter plots we first need to convert the FSPMA object (returned by **fspma.wrapper**) to an average channel object. Note that the example below is the most simple call to this. One has in addition the possibility to use contrasts to get average log channel differences. (see the online help of FSPMA).

```
> fspma.av <- fspma.avchannel(ret)
```

This object can be written into a tab delimited file (the file name is optional to override the default).

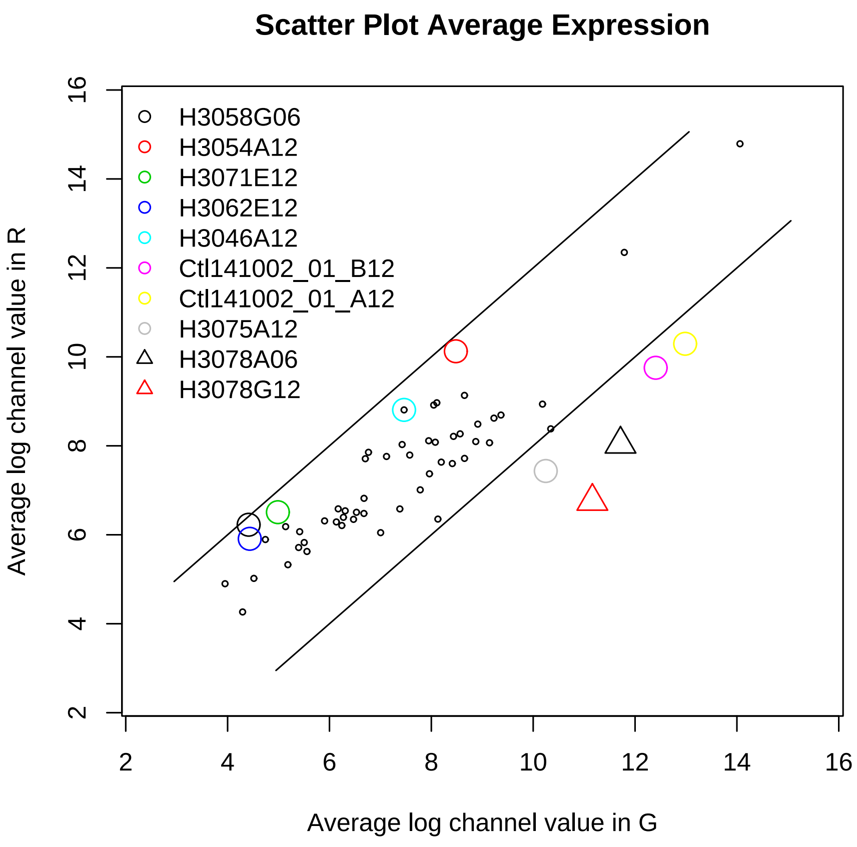
```
> fspma.av.RG2file(fspma.av, filename = "twochannel.RG.av.tsv")
```

The **fspma.av** object can also be used to produce scatter plots of average channel intensities and illustrate those for a specified number of top up amd down regulated and randomly selected “other” genes. This representation is only useful for two channel data since the “G” values are all zero for single channel RG input. The decision about which level is illustrated is made by providing the corresponding rank name (i.e. the identifications we provided in the def file for all levels in the rank effect). They are stored in **ret$info$rank.names** and can, as is illustrated below, be found at the R command line at any time. This call to **fspma.av.scattp** overrides the default amd places the legend to the top left corner.

```
> rank.names <- ret$info$rank.names
> cat(rank.names, sep = ", ")
```

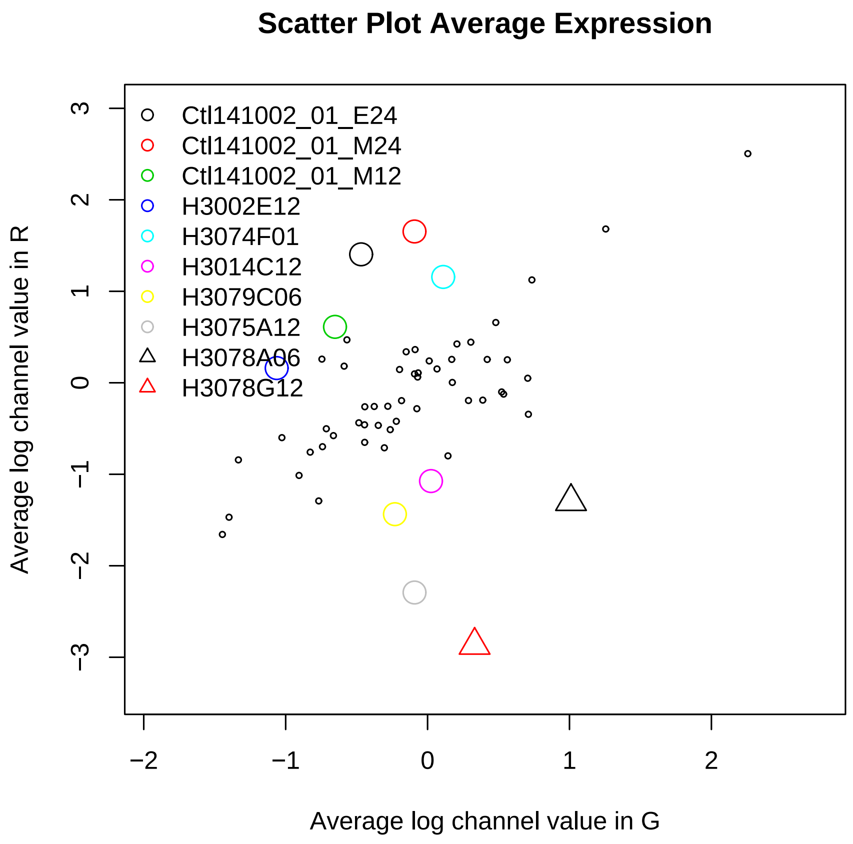
```
adult, day.1, day.5, day.10, day.15, day.23, day.35
```

```
> fspma.av.scattp(fspma.av, "day.1", leg.pos = "topleft")
```



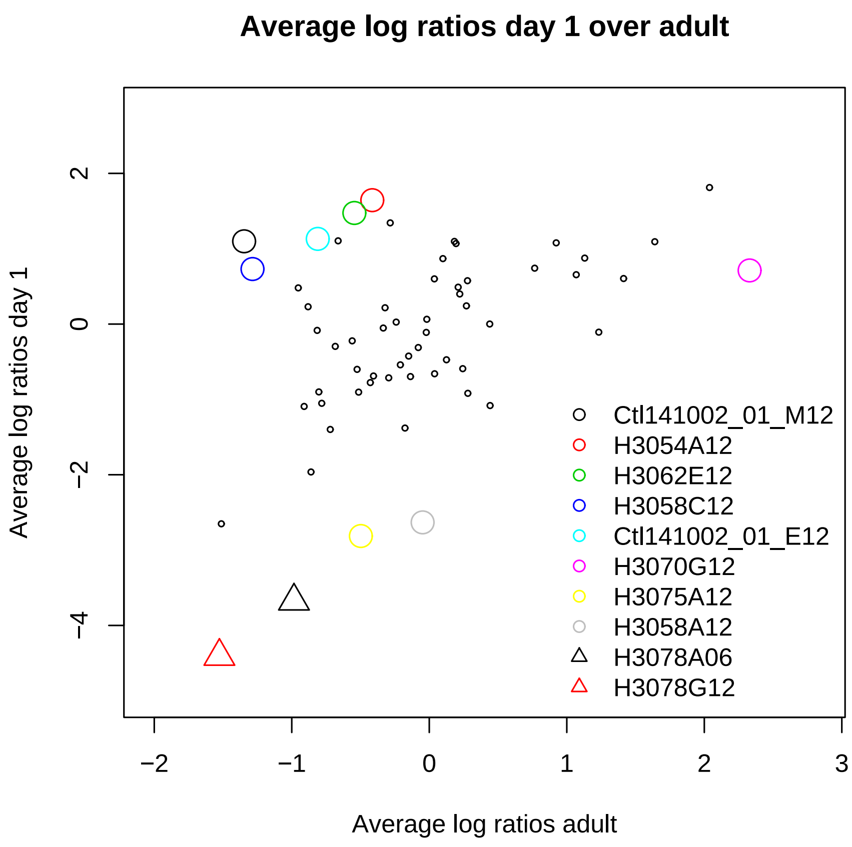
Function **fspma.avchannel** also allows to specify a contrast. In that case we get a plot of log R differences over log G differences. We illustrate this here by plotting the average channel differences of day 1 to day 15 minus day 23 to adult (see also a similar contrast in “twochannel.def”). For plotting these average log channel values, we need to specify the state name ’diff’ (which is the default in this case). Note that this is also an example how we may customise default settings in the plot function. Here we remove the lines that indicate log two fold up and down regulation and put the legend in the top left corner.

```
> fspma.logRGd <- fspma.avchannel(ret, contrast = c(-1, 1, 1, 1,
+     1, -1, -1))
> fspma.av.scattp(fspma.logRGd, "diff", lg.diff = NULL, leg.pos = "topleft")
```



The final option with these functions is to obtain a representation of average log ratio differences between two effects. This representation is again useful for all data sources (i.e. plot log2(R[n]/G[n]) over log2(R[m]/G[m]) for (m,n) being two levels in the rank effect). This is done on the original output of **fspma.avchannel** and requires to specify two state names in the call to **fspma.av.scattp**. Again we customise the plot by providing a different title, x and y labels and remove the lines that bound log two fold up and down regulation.

```
> fspma.av.scattp(fspma.av, c("adult", "day.1"), pl.title = "Average log ratios day
+     x.leg = "Average log ratios adult", y.leg = "Average log ratios day 1",
+     lg.diff = NULL)
```



Further options to **fspma.av.scattp** can be found in the online help of the library.

3.6 Combining FSPMA with other libraries

To allow combining FSPMA with other microarray libraries, we provide two functions that can be used to extract a compatible "RG" object from a FSPMA object and to merge a compatible "RG" object with a FSPMA object. This is done by functions **fspmaRG.2.RG** and **RG.2.fspmaRG**. The next example illustrates this with a new definition file, that was derived from "twochannel.def". For this analysis we specify that **fspma.wrapper** should terminate after having read the data and that we intend to impute missing values with the k nearest neighbour approach suggested in [3]. We thus change the definition file to:

```
# here: set load only flag - fspma.wrapper returns after having read the definition file and data.
Load.Only:      T
# How do we impute: (NA for none, knn <TAB> k for knn using k neighbours
# and del for removing such genes)
Impute.Mthd:    knn     5
```

Then we can take control over the experimental data, before it gets analysed by the library.

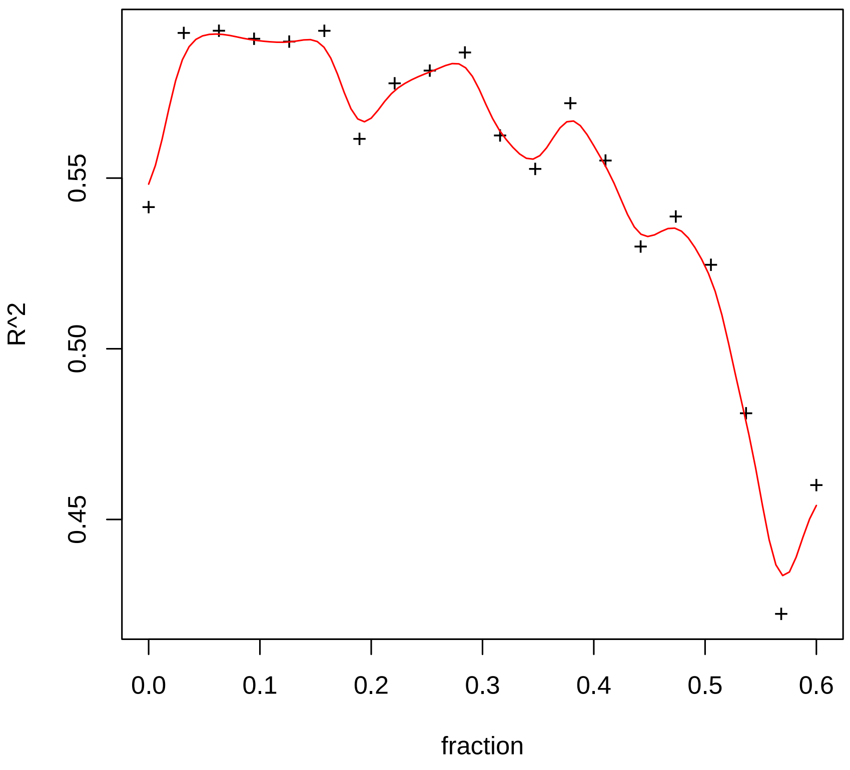
```
> library(fspma)
> ret <- fspma.wrapper("loadonly.def")

Run started on - Mon Jun 12 2006 11:48:50 AM

Loading info from file.
Checking consistency of the definition file loadonly.def was successful.
Loading RG data.
Reading File: R35_NIA1_AWX_ad_612_Fl_output.tsv
Reading File: R35_NIA1_AWX_ad_613_Fl_output.tsv
Reading File: R35_NIA1_AWX_ad_629b_Fl_output.tsv
Reading File: R35_NIA1_AWX_ad_630_Fl_output.tsv
Reading File: R25_NIA1_AWX_d01_s16_Fl_output.tsv
Reading File: R25_NIA1_AWX_d01_s17_Fl_output.tsv
Reading File: R25_NIA1_AWX_d01_s18_Fl_output.tsv
Reading File: R25_NIA1_AWX_d01_s21_Fl_output.tsv
Reading File: R25_NIA1_AWX_d05_s13_Fl_output.tsv
Reading File: R35_NIA1_AWX_d05_545_Fl_output.tsv
Reading File: R35_NIA1_AWX_d05_546_Fl_output.tsv
Reading File: R35_NIA1_AWX_d05_547_Fl_output.tsv
Reading File: R25_NIA1_AWX_d10_s28_Fl_output.tsv
Reading File: R35_NIA1_AWX_d10_596_Fl_output.tsv
Reading File: R35_NIA1_AWX_d10_597_Fl_output.tsv
Reading File: R35_NIA1_AWX_d10_600_Fl_output.tsv
Reading File: R35_NIA1_AWX_d15_594_Fl_output.tsv
Reading File: R35_NIA1_AWX_d15_605_Fl_output.tsv
Reading File: R35_NIA1_AWX_d15_671_Fl_output.tsv
Reading File: R35_NIA1_AWX_d15_674_Fl_output.tsv
Reading File: R35_NIA1_AWX_d23_653_Fl_output.tsv
Reading File: R35_NIA1_AWX_d23_654_Fl_output.tsv
Reading File: R35_NIA1_AWX_d23_655b_Fl_output.tsv
Reading File: R35_NIA1_AWX_d23_670_Fl_output.tsv
Reading File: R35_NIA1_AWX_d35_631_Fl_output.tsv
Reading File: R35_NIA1_AWX_d35_632_Fl_output.tsv
Reading File: R35_NIA1_AWX_d35_633_Fl_output.tsv
Reading File: R35_NIA1_AWX_d35_651_Fl_output.tsv
Number of NA entries from channel:   0.
Processing completed.
```

To use the data loaded by **fspma.wrapper** with functions that know about RG objects, we first have to extract a compatible RG object.

```
> RG <- fspmaRG.2.RG(ret)
```

This allows to use functions that operate on **RG** objects. The example here uses YASMA to obtain a correlation plot over experiments in dependency of removed genes.

```
> rg.rsq.plot(RG)
```



We deduce from this plot that it might be a good idea to remove 5% of low correlated spots. In fact we mark those as NA, and later use FSPMA to deal with this.

```
> RG <- rg.remove.quantile(RG, 0.05, level = 10, set.na = T)
```

This modified RG gets now merged with the FSPMA object.

```
> ret1 <- RG.2.fspmaRG(RG, ret)

Gene names mismatch in fspmaobj resolved.
```

and we finally continue analysis within **fspma.wrapper**, however at this time providing data at the command line. Note that this analysis will take some time. This is due to the k nearest neighbour imputation, we specified above. For full gene sets, we have to expect several hours of runtime!

```
> ret2 <- fspma.wrapper(RG = ret1$RG, info = ret1$info)

Run started on - Mon Jun 12 2006 11:49:06 AM

RG provided data not loaded.
Number of NA entries from channel: 157, see na_debug.tsv for more information.
Number of NA entries from flag: 157 (NA spikes):   0

..................................................
..................................................
..................................................
..................................................
..................................................

..................................................
..................................................
..................................................
..................................................
..................................................
Number of NA entries after imputing:   0
Getting model.info
Do classical normalization.
Normalising by: location
Normalising by: scale
Write raw data to files.
Convert RG data to log ratios.
Calculating ANOVA and write table and variance components.
...Getting number of comparisons
Ranking by Variety.
ANOVA ranking.
Contrast based ranking
base:   1 - shortcut for several contrasts, be patient!
contrast: -0.333333333333333,0.25,0.25,0.25,0.25,-0.333333333333333,-0.333333333333333
Processing completed.
```

3.7 Next Steps

For more information about FSPMA and its definition files it is suggested to study and possibly modify those that come with the library. We also refer to the online help of FSPMA, which provides more insight into all functions, we used here. The online help also discusses functions of FSPMA that are used internally in **fspma.wrapper**. The online help (reference deffile.def) and the other TR which is available under package overview or from FSPMA's web page at `http://www.ccbi.cam.ac.uk/software/psyk/software.html#sykacek_TR051` provide also a reference to all definition file entries.

Chapter 4

FSPMA Reference

`aov.tab.varcomp.2.file`

Calculate ANOVA table and variance components and write them into a file.

Description

Function aov.tab.varcomp.2.file takes a YASMA bfaov object and the fspma's info structure and writes an ANOVA table and the variance components into a file.

Usage

```
aov.tab.varcomp.2.file(av, info, model.info, file=NULL)
```

Arguments

`av`	YASMA's bfaov object
`info`	fspma's info structure
`model.info`	fspma's model.info structure
`file`	optional file name, if NULL the file name is taken from info$aovfname

Value

The function aov.tab.varcomp.2.file takes the bfaov object and some control information from info and model.info to generate the ANOVA table and calculate and write the variance components to a file. The file name in info$aovfname is obtained from the 'ANOVA.Outfile:' entry in the definition file. The function has no return object.

See Also

`tabdel2rg`, `do.normalize`, `do.anova`, `aov.tab.varcomp.2.file` and `contrast.rank.2.file`

Examples

```
## Not run:
## read definition file
info <- read.defs('deffile.def')
## extract model.info from the info structure
model.info <- info.2.yasmaeqns(info)
## read data
RG <- tabdel2rg(info)
## and normalize it
RG <- do.normalize(RG, info)
## convert to array structure
aov.a <- rg2log.array(RG, log.ratio=T)
## and call do.anova
av <- do.anova(model.info, aov.a)
## write the ANOVA and reml variance table according to info
aov.tab.varcomp.2.file(av, info, mode.info)
## End(Not run)
```

`contrast.rank.2.file` *Calculate contrast based within ANOVA rankings and write resulting gene lists to tab delimited files.*

Description

Function contrast.rank.2.file takes a YASMA bfaov object, YASMA's RG structure, fspma's info and model.info structures, calculates the rank lists and writes all information (including the within group sample means) into a results file.

Usage

```
contrast.rank.2.file(av, RG, model.info, info, p.type, p.thrs,
n.comps, log.fname = fspma.logfile)
```

Arguments

`av`	YASMA's bfaov object.
`RG`	YASMA's RG structure.
`model.info`	fspma's model.info structure.
`info`	fspma's info structure.

`p.type`	P-value adjustment (defult from info)
`p.thrs`	Threshold of p-value or top rank counter
`n.comps`	Overall number of comparisons (calculated by fspma.wrapper)
`log.fname`	Name of log file used to dump progress report

Value

contrast.rank.2.file writes rank tables to tab delimited files and returns a list of all rank tables to the calling function. Note that the returned object is not meant to be manipulated at user level.

File structure

The function contrast.rank.2.file takes the bfaov object and some control information to generate contrast based rank tables within the ANOVA. Each comparison is written in its own tab delimited file. The files contains the index of the gene in the gene list, the gene name, an adjusted p-value, a boolean indicator of upregulation and sample means in each level of th erank effect. Upregulation is decided based on the sign of the comparison. If it specifies a base level, upregulation indicates a significantly larger (differential) expression in the second level of the comparison. In case of a contrast, upregulation indicates that the average differential expression at the levels that have positive weight is higher than the average differential expression at the levels that have negative weight (see also the example below).

The function is controlled by info$contrast, info$p.type and info$p.thrs. The file name base is specified as info$ctrfname, it represents the file name in the 'Contrast.Outfile:' line in the definition file. The component info$contrast is obtained from the 'Base.Contrast:' entry in the definition file. The name part of the contrast entry (first string after tabulator) is concattenated to the file base name to make a unique augmented file base.

Base Levels - Multiple Pairwise Comparisons

There are two different ways to specify a contrast: First one can provide an index of a base level of the rank effect. This base level is compared to all other levels of the effect (e.g. for comparing adult against all other generations in a time course). In this case there is one ranking obtained for every level in the rank effect that differs from the base level. Each ranking is stored in a separate file that starts with "test." followed by the level name and the augmented file base to give a unique file name. Ranking is based on the corresponding p-values that are adjusted based on the overall number of comarisons suggested by the definition file, which is vital to get less false positives.

Contrasts

The other option requires as many entries as there are levels to specify a contrast directly. If the contrasts do not sum to zero, positive and negative contrasts are normalized separately. In a mammary gland study with entries 'day 0 lactation', 'day 5 lactation', 'day 10 lactation', '12 hrs involution', '24 hrs involution', '72 hrs involution' and '96 hrs involution' we can thus use the contrast -1<TAB>-1<TAB>-1<TAB>-1<TAB>1<TAB>1<TAB>1 to specify a contrast that ranks w.r.t. a type 2 apoptosis event. (The values are normalized to -1/4 and 1/3 respectively and upregulation "TRUE" would denote those genes that are upregulated at late involution time points). The file name is generated by the contrast name that is concatenated with the file base specified as 'Contrast.Outfile:'.

Controling Significance

The component info$p.type is obtained from the 'Comp.Type:' entry that can specify various adjustments (see p.adjust in R or `deffile.def` for help). The number of comparisons to be done within one analysis run (i.e. specified within a definition file) is used as counts to adjust the p-values.

The component info$p.thrs is obtained from the 'Comp.Thrs:' entry that either specifies a p-value threshold or if it is an integer number a count of how many of the most significant genes should be written into the file.

See Also

`tabdel2rg`, `do.normalize`, `do.anova`, `deffile.def` and `aov.tab.varcomp.2.file`

Examples

```
## Not run:
## read definition file
info <- read.defs('deffile.def')
## extract model.info from the info structure
model.info <- info.2.yasmaeqns(info)
## read data
RG <- tabdel2rg(info)
## normalize it
RG <- do.normalize(RG, info)
## convert to array structure
aov.a <- rg2log.array(RG, log.ratio=T)
## call do.anova
av <- do.anova(model.info, aov.a)
## and calculate and write within ANOVA rank tables.
contrast.rank.2.file(av, RG, model.info, info, n.comps=[plug in here!])
## End(Not run)
```

`deffile.def` *Example definition file for fspma.wrapper*

Description

Definition file that controls the execution of the fspma scripts.

Usage

```
deffile.def
```

Format

deffile.def is a file name that refers to a tab delimited definition file which controls all aspects of evaluating a miroarray experiment. It is used as parameter to `fspma.wrapper`(deffile), see `fspma.wrapper`.

Details

The tags at the beginning of each line identify the information fspma uses to control the analysis of a microarray experiment. It is of vital importance that they are typed exactly, since they are searched for. Lines starting with # are regarded as comments. All different specifiers within a line (e.g. the numbers of instances for each effect in the 'Effects:' row) must be separated by Tabulators. Avoid at every cost extra (trailing!) white spaces (blanks, tabulators, etc.), because these will lead to completely undesired behaviour! The following provides an example definition file. For a detailed discussions of various options refer to the man pages of individual functions. Also refer to the examples provided with the library and the online example at `http://www.ccbi.cam.ac.uk/software/psyk/software.html#fspma` that can be used in connection with a dataset that is available online.

See Also

`do.normalize` and `spike.normalize` for further discussion of available normalization options, `contrast.rank.2.file` for an explanation of all possibilities with the contrast entries, `specify.experiments` for how to specify different experiments and `fspma.wrapper` on how to get the analysis started.

Examples

```
## Not run:
# Definition file for ANOVA analysis of an experiment.
# This file is the interface of any experiment that can be analysed
# by YASMA. It is read by an r-script that loads data and prepares all
# what is needed afterwards during the analysis.
```

```
#
# (C) P. Sykacek 2004.
#
# Version information: This will be counted up if changes are made to
# the structure of the definition file, to be able to parse old files as
# well. Note that the description of old files is no longer available,
# as soon as this gets updated. (i.e. version 1 is obsolete by now)
Ver:    2
#
ExpName:        Mouse Testis
#
# All effects that appear in the experiment. The last is the
# minimum number of replicates per slide and must be specified.
Effects:        7       4       2
#
# Experiment is Unbalanced: (Otherwise an error is raised if the
# experiment is unbalanced!) Unbalanced experiments can be loade and
# normalized. An ANOVA analysis however is not possible.
Is.Unbalanced:  F
#
# Effect Names: (compatible with R variable names, no blanks and underscore)
Eff.Nams:       time    sample  rep
#
# which are random effects
RandEffs:       F       T       T
#
# Which effect should be ranked over?
Rank.Eff:       1
#
# Names of each ranking dimension that need to be compatible with R variable
# names. Hence no blanks and underscores.
Rank.Names:     adult   day.1   day.5   day.10  day.15  day.23  day.35
#
# The following line either specifies a contrast or a base level to be used for
# ranking within the ANOVA model.
# There are 3 possibilities: specify a contrast; specify a base level
# of the rank effect that all other levels should be compared against;
# specify ANOVA based ranking, where the p-value of each time course (or the
# respective type) is evaluated with an ANOVA model.
# Novel: since we allow for multiple comparisons in one file we add
# a name for each contrast that enters the file name.
# Example for a contrast comparing day 15 versus adult
# Base.Contrast:        ctadd15 -1      0       0       0       1       0       0
# Example for general ranking against adult (takes most significant p-value
# for each gene)
# Base.Contrast:        blad    1
# If there is only one level in the rank effect the previous line ranks
# based on significant differential expression between the two channels.
# Example for ANOVA based ranking:
```

```
# Base.Contrast:        ANOVA
# Example for VARIETY based ranking. Only for two channel arrays
# where this compares the average log ratios against each other:
# Base.Contrast:        VARIETY
# Several Base.Contrast: entries are possible. They will all be analysed
# together with p-values adjusted accordingly.
# The following example specifies two base levels and does hence 12
# pairwise comparisons for every gene. This is one replicated, which
# will lead to a conservative p-value adjustment. This can be avoided by
# using one base level and specify the remaining 5 pairwise comparisons
# by conventional contratsts.
Base.Contrast:  blad    1
Base.Contrast:  bld01   2
# Output file to write the gene list from the comparison,
# use NA if not required
Contrast.Outfile:       testis_gen_comp.tsv
# Type of p-value adjustment: holm, hochberg, hommel, bonferroni, fdr,
# none (from R{stats} p.adjust)
Comp.Type:      bonferroni
# p-value threshold or integer number interpreted as top-ranked count.
Comp.Thrs:      0.05
# ANOVA table and variance component output file
ANOVA.Outfile:  mouse_testis_aov.txt
#
# Normalization control; NA if none otherwise loess,
# location or scale combinations like loess scale or
# location scale are allowed as well. For loess scale the order matters!
# This is an interface to YASMA's functionality.
# loess with optional span and degree (0.25< span <=1 and degree 1 or 2)
# default is 0.9 for the span and 1 for the degree.
# Normalization:        loess   [<span> [<degree>]]
# IMPORTANT: Without spike genes, loess is invalid for single channel arrays!
# Normalization:        localtion
# Normalization:        location        scale
# Normalization:        loess   scale
# Version 2 allows spike based normalization which is initialised using
# the modifier spike. e.g.
# Normalization:        spike   location
# In addition to location and scale more elaborate versions based on
# grid position and amplitude are possible. The latter fits a loess fit
# to the spikes residuals and subtracts that from all other gene values.
# (which depending on the data, are either log ratios or log expressions).
# possible values are spatial and amplitude e.g.
# Normalization:        spike   spatial amplitude
# Spike normalization can be modified by grid and scale and by two
# numerical values which are taken as loess span and degree. Modifier grid
# uses grid numbers as factors in the loess model equation and allows to
# remove pin effects. Spike based normalization requires a list of
# spikes and expected expressions to be provided after spike.list: (see below)
# The following is a possible setting for spike based normalization.
# It uses a loess fit with spatial position (spatial effects) and
# amplitude (spot intensity) as continuous regressors and the grid
# index (pin effects) as factorial regressor. The optional span and
# degree of the loess fit are here set to 0.95 and 1 respectively.
# After calculating the overall scale, each slide is
# transformed to that scale.
Normalization:  spike   0.95    1       spatial amplitude       grid    scale
#
# Control conversion; RG -> array (T -> take log, F -> use data as is)
DoRG2logarray:  T
#
# Value for flagged entries; If available checked against Flag. column
# The flag value can be numeric or string and there can be more than one
# value in this line, separated by tabulators.
# For bluefuse manual exclusion:
# Flag.NOK:     yes
# For genepix : (both automatic and manually excluded flags)
# Flag.NOK:     -50     -100
Flag.NOK:       NA
#
# Select method to impute NA values. This is needed in connection with
# Flags to deal with missing values.
# Remove genes with missing spots from analysis:
# Impute.Mthd:  del
# Impute with knn (Trojanskaya's Bioinformatics publication), k is an
# integer number (the number of neighbours that are considered):
# Impute.Mthd:  knn     k
# Note that in case we find missing spots and NA was selected,
# the script issues a waning and changes to "del". Otherwise the
# subsequent analysis will crash.
# No imputation:
Impute.Mthd:    NA
#
# Load.Only allows to control the program flow in the script. If T
# The script aborts after loading the data before anything else has
# been done with the data. Together with an alternative entry,
# this gives the user some flexibility to process data in a more
# flexible way by combining the automatic analysis with other R-code.
Load.Only:      F
#
# The following entry allows to write the normalized log ratios to an output file.
# NA is for none. The string is actually only the base file name with two
# endings. In this case 'nia_rawlogrt.tsv' contains the data that was
# processed acording to the normalization and log options. Data is
# stored row wise (i.e. each full set of genes is one row) with the gene
# symbols used as colum names. The second file has as many rows as the
# previous, such that each row identifies the level settings of the
# corresponding data row. In this case it would be named 'nia_raweffdesc.tsv'.
```

```
DataOutFnam:    nia_raw
#
# The following definitions allow to identify the various columns that
# are extracted from the tab delimited microarray files.
# column of gene name and its name in the header
gene.colnam:    NAME
#
# column of Cy5 spot and its name in the header
R.colnam:       AMPCH1
#
# column of Cy5 background and its name in the header: NA if unavailable.
Rb.colnam:      NA
#
# column of Cy3 spot and its name in the header: may be missing (NA)
G.colnam:       AMPCH2
#
# column of Cy3 background and its name in the header
Gb.colnam:      NA
#
# column of flag id and its name in the header
Flag.colnam:    NA
#
# X-pos column name (NA if none)
X.colnam:       PELCOL
#
# Y-pos column name (NA if none)
Y.colnam:       PELROW
#
# Grid.no. column name (NA if none)
Grid.colnam:    BLOCK
#
# Do a fast file load? If one sets load.fast: to TRUE, fspma assumes that
# the positions of genes found in the first file is identical with the
# gene positions in all subsequent files. This speeds up data loading
# considerably. If there is any doubt about whether the gene positions
# are identical, one should always set this flag "FALSE".
load.fast:      F
#
# Next: allocate files to experiments. The number is obtained automatically from
# the effects description. First column: file name , then no effects minus 1
# columns to allocate the data and as final column a Boolean flag (T or F)
# indicating whether the corresponding file contains a dye swap.
# Unless one states that Is.Unbalanced: T (is true), all levels of
# the effects have to be allocated. Missing specifications lead to an
# error message and processing terminates.
file.alloc:
# adult generation                     time    sample  dye swap?
R35_NIA1_AWX_ad_612_Fl_output.xls       1       1       F
R35_NIA1_AWX_ad_613_Fl_output.xls       1       2       F
```

```
R35_NIA1_AWX_ad_629b_Fl_output.xls      1       3       F
R35_NIA1_AWX_ad_630_Fl_output.xls       1       4       F
# day one
R25_NIA1_AWX_d01_s16_Fl_output.xls      2       1       F
R25_NIA1_AWX_d01_s17_Fl_output.xls      2       2       F
R25_NIA1_AWX_d01_s18_Fl_output.xls      2       3       F
R25_NIA1_AWX_d01_s21_Fl_output.xls      2       4       F
# day 5
R25_NIA1_AWX_d05_s13_Fl_output.xls      3       1       F
R35_NIA1_AWX_d05_545_Fl_output.xls      3       2       F
R35_NIA1_AWX_d05_546_Fl_output.xls      3       3       F
R35_NIA1_AWX_d05_547_Fl_output.xls      3       4       F
# day 10
R25_NIA1_AWX_d10_s28_Fl_output.xls      4       1       F
R35_NIA1_AWX_d10_596_Fl_output.xls      4       2       F
R35_NIA1_AWX_d10_597_Fl_output.xls      4       3       F
R35_NIA1_AWX_d10_600_Fl_output.xls      4       4       F
# day 15
R35_NIA1_AWX_d15_594_Fl_output.xls      5       1       F
R35_NIA1_AWX_d15_605_Fl_output.xls      5       2       F
R35_NIA1_AWX_d15_671_Fl_output.xls      5       3       F
R35_NIA1_AWX_d15_674_Fl_output.xls      5       4       F
# day 23
R35_NIA1_AWX_d23 653_Fl_output.xls      6       1       F
R35_NIA1_AWX_d23 654_Fl_output.xls      6       2       F
R35_NIA1_AWX_d23 655b_Fl_output.xls     6       3       F
R35_NIA1_AWX_d23 670_Fl_output.xls      6       4       F
# day 35
R35_NIA1_AWX_d35_631_Fl_output.xls      7       1       F
R35_NIA1_AWX_d35_632_Fl_output.xls      7       2       F
R35_NIA1_AWX_d35_633_Fl_output.xls      7       3       F
R35_NIA1_AWX_d35_651_Fl_output.xls      7       4       F
#
# Provide the spike list with
# name<TAB>R_concentration<TAB>G_concentration. Note that
# R and G refer to the above channel names and are the
# known spike concentration in that channel (only the
# correct ratio is important). Hence 1<TAB>1 would also
# be apropriate for non differentially expressed housekeeping genes.
spike.list:
#                       R_con.  G_con.
Ctl141002_01_A01        1       1
Ctl141002_01_A02        1       1
Ctl141002_01_A03        1       1
Ctl141002_01_A04        1       1
Ctl141002_01_A05        1       1
Ctl141002_01_A06        1       1
Ctl141002_01_A07        1       1
#... and more if available
```

```
# Finally a list of all unique gene names (as found in gene.col in the data).
# That should go into the experiment. This is necessary to allow for
# experiment designs where people use different numbers of replicates
# on slide.
genes.list:
H3078A06
H3078C06
H3078E06
H3078G06
H3078A12
#... and many more gene id.'s (all are required!)
## End(Not run)
```

`do.anova` *Interface to yasmas bfaov function (generate ANOVA object)*

Description

Generate the YASMA ANOVA object that is further used to print ANOVA tables, calculate and print variance components and to calculate and print rank tests.

Usage

```
do.anova(model.info, array.data)
```

Arguments

`model.info`	Standard model equations and additional information to generate the ANOVA object.
`array.data`	Array representation of a balanced design obtained by YASMA's rg2log.array or rg2array functions (See yasma help for additional information).

Value

The function returns a bfaov object (See YASMA for details).

See Also

`tabdel2rgdo.normalize`, `aov.tab.varcomp.2.file` and `contrast.rank.2.file`

Examples

```
## Not run:
## read definition file
info <- read.defs('deffile.def')
## extract model.info from the info structure
model.info <- info.2.yasmaeqns(info)
## read data
RG <- tabdel2rg(info)
## and normalize it
RG <- do.normalize(RG, info)
## convert to array structure
aov.a <- rg2log.array(RG, log.ratio=T)
## and call do.anova
av <- do.anova(model.info, aov.a)
## End(Not run)
```

`do.normalize` *Controls the normalization procedure in fspma.wrapper.*

Description

Some simple slide normalization procedures can be selected via the definition file. These include within slide location (i.e. mean) removal, within slide loess normalization, scale removal (i.e. convert log ratios within slide to unit std. deviation) or none, if normalization is done as an extra preprocessing step.

Usage

```
do.normalize(RG, info, log.fname=fspma.logfile)
fspma.logRG.loess.norm(RG, deg=1, span=0.6, show=T, show.new=F, flat=F, tolog=F,..
fspma.logRG.shift.norm(RG,method=c("mean","median"), tolog=F)
fspma.logRG.scale.norm(RG,method=c("sd","mad",), tolog=F)
```

Arguments

`RG`	data structure as described in YASMA (and SMA)
`info`	control structure as obtained by `read.defs`
`log.fname`	Log file name to dump status messages, defaults to fspma.logfile
`deg`	polynomial degree of loess fit
`span`	span of loess fit
`show`	boolean flag to control graphical illustration
`show.new`	boolean flag to control graphical illustration (after normalisation)

`flat`	boolean flag to control graphical illustration (if TRUE, plot base line instead of loess curve in figure)
`method`	specify how the location (for shift) or scale of the data are obtained
`tolog`	boolean flag that decides whether data is moved to log scale or whether it is already on log scale.
`...`	Parameters handed through to plot function.

Details

Normalization is controlled by info$norm.type, a vector that contains the control statements in the "Normalization:" line in the definition file. Valid entries are NA, loess, location, scale and combinations like location<TAB>scale and loess<TAB>scale. For loess one can optionally specify a span (between 0.25 and 1) and a degree (1 or 2). The function returns RG, the experiments RG structure with all data normalized within slides. **Without spike genes, loess is invalid for all single channel specifications!** Consequently this setting is not accepted. FSPMA provides slightly modified versions of YASMAs original normalisation functions (`fspma.logRG.loess.norm`, `fspma.logRG.shift.norm` and `fspma.logRG.scale.norm`). The essential difference is that FSMPA must cope with data that is on log scale. This changes how data is moved to the log ratio - amplitide representation.

Value

All functions return a normalised RG object.

See Also

`fspma.wrapper` and `spike.normalize`

Examples

```
## Not run:
## read definition file
info <- read.defs('deffile.def')
## read data
RG <- tabdel2rg(info)
## and normalize it
RG <- do.normalize(RG, info)
## End(Not run)
```

`do.rank.variety` *Variety based ranking of time course data (or other multi-level effects)*

Description

"Variety" based ranking for significant differences of average channei expression over the entire experiment.

Usage

```
do.rank.variety(av, RG, model.info, info, n.comps)
```

Arguments

`av`	YASMA's bfaov object.
`RG`	YASMA's RG structure.
`model.info`	fspma's model.info structure.
`info`	fspma's info structure.
`n.comps`	Overall number of comparisons (calculated by fspma.wrapper over all comparisons specified in the definition file.)

Details

Variety based ranking is available with the standard YASMA package. The function do.rank.variety wraps around a version in YASMA that is based on within ANOVA ranking, using t-distributions. This approach is only useful for dual channel reference designs, where one tryes to assess significance between the two groups that make up the RNA mix of each channel. Ranking is based on YASMA's bfaov object that is used to extract all relevant information. The null hypothesis is zero log ratios. The rank table contains gene indices, names, adjusted p-values, an up-regulation flag and the average log ratio. It is written tab delimited to a file that concatenates 'test.variety.' with the name specified as info$ctrfname (the 'Contrast.Outfile:' entry in the definition file). Although it will produce outputs for single channel arrays, this procedure makes conceptually no sense in that case!

Value

do.rank.variety provides a ranking that assesses the average channel log ratios for significance. A rank table is written as tab delimited file and returned as object to the calling function. Note that this object is not intended to be modified at user level.

See Also

tabdel2rg, do.normalize, do.anova, aov.tab.varcomp.2.file link{fspma.aovrnk} and contrast.rank.2.file

Examples

```
## Not run:
## read definition file
info <- read.defs('deffile.def')
## extract model.info from the info structure
model.info <- info.2.yasmaeqns(info)
## read data
RG <- tabdel2rg(info)
## and normalize it
RG <- do.normalize(RG, info)
## convert to array structure
aov.a <- rg2log.array(RG, log.ratio=T)
## and call do.anova
av <- do.anova(model.info, aov.a)
## calculate and write ANOVA rank tables (one ANOVA per gene).
do.rank.variety(av, RG, model.info, info, n.comps=[plug in here])
## End(Not run)
```

`file.out.RG` *Write a microarray experiment into a tab delimited file*

Description

This function is provided to convert reference design experiments from various existing data structures to a standardised tab delimited output file format.

Usage

```
file.out.RG(info, RG, wrtchn=F)
```

Arguments

`info`	A fspma info structure
`RG`	A YASMA RG object that is written as tab delimited file.
`wrtchn`	A boolean flag to control whether we write log ratios (default) or log channel information (set wrtchn=TRUE).

Details

This function is provided to convert reference design experiments from various existing data structures to a standardised output file format. The function writes two files, both names start as is specified in info$datoutfname (entry DataOutFnam: in the definition file). Depending on wrtchn, the first file(s) store(s) either log ratios between Cy5 and Cy3 and thus ends with 'logrt.dat', or there are two files with log channel values which end with 'logR.dat' and 'logG.dat'. The function also stores the classification information (i.e. the grouping information) of the experiment and is stored in the file ending with 'effdesc.dat'.

Value

The function writes the experiments log ratios and the corresponding effect values into tab delimited files that can be used as input to other processing tools.

See Also

`tabdel2rg`, `deffile.def`, `fspma.wrapper`,

Examples

```
## Not run:
# read the data according to the definition in info
RG <- tabdel2rg(info)
# write log ratios and corresponding effect values as tab delimited
# files
file.out.RG(info, RG)
## End(Not run)
```

`fspma.aovrnk` *ANOVA based ranking of time courses (or other multi-level effects)*

Description

ANOVA based ranking of experiments with respect to multi-level effects like time courses.

Usage

```
fspma.aovrnk(info, model.info, array.data, fname=NULL, n.comps)
```

Arguments

`model.info`	fspma's model.info structure.
`info`	fspma's info structure.
`array.data`	Array representation of a balanced design obtained by YASMA's rg2log.array or rg2array functions (See yasma help for additional information).
`fname`	Optional output file name, default taken from info structure (info$ctrfname which resembles the 'Contrast.Outfile:' entry in the definition file). To guarantee unique file names this name is augmented with the string "anova." at the beginning.
`n.comps`	Number of comparisons to adjust for (this is calculated for the entire definition file).

Details

The ranking is based on the p-value of an ANOVA model calculated for each gene. The null hypothesis is that all levels in the rank effect have the same mean. Calculations are based on YASMA's bfaov object that is internally used to calculate the p-values of mixed models. Information about random effects etc. is treated apropriately. The function generates a tab delimited file that contains the gene information, the p-value of the F-test and the sample averages of all levels of the rank effect.

Value

fspma.aovrnk provides a ranking by means of single gene ANOVA models. The rank table is written as tab delimited file and returned to the calling function. Note that this object is not meant to be manipulated at user level.

See Also

`tabdel2rg`, `do.normalize`, `do.anova`, `aov.tab.varcomp.2.file` and `contrast.rank.2.file`

Examples

```
## Not run:
## read definition file
info <- read.defs('deffile.def')
## extract model.info from the info structure
model.info <- info.2.yasmaeqns(info)
## read data
RG <- tabdel2rg(info)
## and normalize it
RG <- do.normalize(RG, info)
## convert to array structure
aov.a <- rg2log.array(RG, log.ratio=T)
```

```
## and call do.anova
av <- do.anova(model.info, aov.a)
## calculate and write ANOVA rank tables (one ANOVA per gene).
fspma.aovrnk(info, model.info, array.data, n.comps=[plug in here])
## End(Not run)
```

`fspma.avchannel` *Conversion of FSPMA objects to average channel expressions, storage and visualisation*

Description

These functions extract average channel expressions and derived information from FSPMA objects. They allow to store and plot the resulting representation.

Usage

```
fspma.avchannel(fspma.obj, contrast=NULL, statenames=NULL, log2exp=F)
fspma.av.RG2file(fspma.av.RG, filename='average.channels.tsv')
fspma.av.scattp(fspma.av.RG, statename, filename='NA',
pl.title='Scatter Plot Average Expression',
x.leg='Average log channel value in G',
y.leg='Average log channel value in R',
nsamples=50, lg.diff=2, ndiff.smpls=5, plt.lwd=1,
n.sz=0.5, ud.sz=2, col.frst=T, p.dim=7.0, leg.pos='bottomright',
leg.bty='n')
```

Arguments

`fspma.obj`	is either a FSPMA object as generated by fspma.wrapper
`contrast`	Optional contrast definition used to obtain channel averages over subsets of rank levels. These contrasts are of identical length and striture as those specified in the definition file.
`statenames`	Optional state names. Default are the rank names from the definition file and ”diff” for average expression according to a contrast.
`log2exp`	Boolean control whether we exponentiate the averages. Defaults to FALSE.
`fspma.av.RG`	An average channel object that is obtained by calling fspma.avchannel.
`filename`	In fspma.av.RG2file: a file name that is used to store the average expressions in a tab delimited format with ”average.channels.tsv” as default value. In fspma.av.scattp: a file name to store the figure as encapsulated postscript or ’new’ if a new graphics window should be used. In the default case of ’NA’, the current ”device” is used for plotting.

`statename`	Name of the level one intends to visualise in the scatter plot. In the default case this is either a rank name or ”diff”.
`pl.title`	Plot Title.
`x.leg`	X axis legend.
`y.leg`	Y axis legend.
`nsamples`	Number of subsampled ”non suspicious” genes to be plotted.
`lg.diff`	Average log fold indication lines (NA for none).
`ndiff.smpls`	Number of suspicious genes to be plotted (those with largest up and down regulation).
`plt.lwd`	Line width in plot.
`n.sz`	Point size for non suspicious genes.
`ud.sz`	Point size for suspicious genes.
`col.frst`	Boolean flag to control whether colour or symbols should alter first.
`p.dim`	Figure dimension in postscript device.
`leg.pos`	Position of legend (see plot legend function).
`leg.bty`	Type of legend bounding box (see plot legend function).

Value

`fspma.avchannel` converts a FSPMA object to a fspma.av.RG object which stores average channel expression values. The other functions have no return value. They write information to files or plots.

See Also

`fspma.wrapper` and `fspma.rankplot`

Examples

```
## Not run:
fspma.obj <- fspma.wrapper('twochannel.def')
fspma.av <- fspma.avchannel(fspma.obj)
allnams <- fspma.obj$info$rank.names
fspma.av.scattp(fspma.av.RG, allnams[1])
## End(Not run)
```

fspma.rankplot *Data visualisation, FSPMA rank results and M/A Plots*

Description

These functions are useful tu visualise data that should be analysed by FSPMA. We provide rank plots and plots that visualise differential expression over spot amplitude.

Usage

```
fspma.maplot(fspma.obj, slideno=1, pl.file='NA', pl.title='M-A Plot',
have.spike='no', nsmpls=50, xlab='Log Amplitude',
ylab='Log Ratio', cex=0.5, col=2, leg.pos='bottomright', leg.bty='n')
fspma.rankplot(fspma.obj, list.nam, no.genes=5, pl.title='Rank List',
pl.file='NA', plt.lwd=2, col.frst=T, leg.pos='bottomright', leg.bty='n')
```

Arguments

`fspma.obj`	is either a FSPMA object as generated by `fspma.wrapper`
`slideno`	number of slide that should be plotted in M/A representation
`pl.file`	a file name to store the figure as encapsulated postscript or 'new' if a new graphics window should be used. In the default case of 'NA', the current "device" is used for plotting.
`pl.title`	Plot Title.
`have.spike`	Spike control: 'no' - spike genes are not plotted. 'yes' spike genes are plotted if available.
`nsmpls`	Number of subsampled genes to be plotted.
`xlab`	X axis legend.
`ylab`	Y axis legend.
`cex`	Plot symbol size.
`col`	Point colour.
`leg.pos`	Position of legend (see plot legend function).
`leg.bty`	Type of legend bounding box (see plot legend function).
`list.nam`	Name of rank list to be visualised. The names replicate the names used in fspma.wrapper to write rank tables to files.
`no.genes`	Number of top ranked genes to be visualised.
`plt.lwd`	Point width of lines in the plot
`col.frst`	Boolean indicator whether to alter colors first and then symbols.

Value

Neither function returns values. Both plot information.

See Also

`fspma.wrapper` and `fspma.avchannel`

Examples

```
## Not run:
fspma.obj <- fspma.wrapper('twochannel.def')
fspma.maplot(fspma.obj)
rank.tab.nams <- names(fspma.obj$plt.tab)
fspma.rankplot(fspma.obj, rank.tab.nams[1])
## End(Not run)
```

`fspma.set.globals` *FSPMA globals and handling function*

Description

Global variables that are used internaly to control FSPMA's functionality and handler function.

Usage

```
fspma.eps
fspma.knn.max.tol
fspma.logfile
fspma.set.globals(this.eps=10^-10, this.knn.max.tol=0.001,
    this.logfile='fspma.log.txt')
```

Arguments

`this.eps`	eps value written to fspma.eps. The latter is used in FSPMA as lower bound of expression values before moving to log 2 scale.
`this.knn.max.tol`	maximum tolerance limit written to fspma.lnn.max.tol. The latter controls the number of iterations in knn imputing.
`this.logfile`	file name copied to fspma.logfile. The lattre is used as sink to write error messages and the processing status.

Value

The function has no value but writes to FSPMA global variables.

Examples

```
## Not run:
## set a different log file
fspma.set.globals(this.logfile='mylog.txt')
## which is now used by fspma.wrapper to write the processing status.
ret <- fspma.wrapper('myexperiment.def')
## reset to fspma defaults
fspma.set.globals()
## End(Not run)
```

`fspma.wrapper` *Friendly Statistics Package for Microarray Analysis (fspma) or Stats' without tears (i.e. coding)*

Description

Microarray data analysis with normalisation, bad quality expression treatment, ANOVA analysis and within ANOVA tests controlled by definition files.

Usage

```
fspma.wrapper(def.fname=NULL, RG=NULL, info=NULL,
na.debug.fname='na_debug.tsv', log.fname=fspma.logfile, init.logfile=T)
```

Arguments

`def.fname`	File name of a definition file that controls analysis.
`RG`	An optional (extended) YASMA RG object that, if provided, will skip the data loading step and increase the efficiency of multiple rankings of the same data (e.g. using different normalisation methods). This RG object is an extension of yasma's RG structures and so far kept downward compatible. Some functions of yasma which modify RG (like normalization) tend to remove parts they do not know about. Calling such code will lead to incompatibilities of the resulting RG object with some functions within this library (e.g spike based normalization)! This can be resolved by using the RG adaptors provided by the library (`fspmaRG.2.RG`).
`info`	AN optional info structure otherwise read from definition file. This allows together with the Load.Only: entry in the definition file to include other processing steps on a scripting level and to repeat processing.
`na.debug.fname`	An optional parameter that is used to write debug information if NA values are read from the file (i.e. before the flaged spots are treated).

`log.fname` Name of an ASCII log file used to store information about analysis steps. The name defaults to "fspma.log.txt". Use the log file to resolve problems with definition files.

`init.logfile`
Optional boolean flag that controls whether a definition file should be newly initialised. Defaulst to TRUE.

Details

fspma.wrapper is an efficient interface to microarray analysis with the YASMA package that comes with several generalizations to allow data conversion, normalization, ANOVA and test based inference for rather general microarray experiments. The function is controlled via a definition file that is adaptable to every balanced reference design and slide data. It covers two colour arrays with and without background signals and one colour data like affymetrix and meta data generated thereof, that was normalized or pre-processed elsewhere. The main restriction of this script is that it requires all slide-files of one experiment to use coherent column names in all files. If your experiment violates this rather modest requirement, then you need to process your files prior to using fspma. Details about definition files are provided in `deffile.def` for a time course with technical replicates and on slide replication. Some working examples are provided with the documentation as well. Finally we also povide a definition file with download instructions on how to obtain the accompanying arrays of a publicly available Affymetrix dataset at `http://www.ccbi.cam.ac.uk/software/psyk/software.html#fspma`.

For further information on info see `read.defs`, for information on model.info see `gen.yasma.eqns`. RG, aov.a and av are yasma's RG structure, yasma's array data and yasma's ANOVA object and described in more detail in the help to the yasma package.

Value

fspma.wrapper returns a fspma object. The object summarises all information that resulted from the analysis run. In particular we have:

fspma.obj$RG - an extended RG structure

fspma.obj$info - R representation of the definition file

fspma.obj$model.info - all model equations (spike normalisation, ANOVA model and models required for ranking and p-value calculation.

fspma.obj$aov.a and fspma.obj$av - YASMA anova objects for debuging purpouse.

fspma.obj$plt.tabs - all rank tables useful for visualisation purpouse.

If the "load only" flag (info$load.only which corresponds to the Load.Only: entry) was set in the definition file, fspma.obj contains only the first two elements.

See Also

tabdel2rg, do.normalize, spike.normalize, do.anova, fspmaRG.2.RG, aov.tab.varcomp.2.file
contrast.rank.2.file, fspma.rankplot and fspma.avchannel

Examples

```
## Not run:
 ## FSPMA's logic is in the definition file. Analysis starting with
 ## data loading, normalisation, bad quality flag treatment, ANOVA
 ## calculations and rank tables are obtained in one go.
 ret <- fspma.wrapper('mouse_testis.def')
 ## And later, as soon as you have more understanding of what's going
 ## on inside, you get more flexibility with a def file that
 ## sets Load.Only T
 ret <- fspma.wrapper('mouse_load_only.def')
 RG <- fspmaRG.2.RG(ret)
 ## now use yasma functions on RG ( e.g. here we plot correlation
 ## over quantile of removed data)
 rg.rsq.plot(RG)
 ## and reduce the dataset manually (remove 0.1 quantile of least
 ## correlated genes,  which was found to be a sensible number by the
 ## previous plot)
 RG <- rg.remove.quantile(RG, 0.1)
 ## now back to FSPMA data:
 ret.new <- RG.2.fspmaRG(RG, ret)
 ## before we continue with the fspma.wrapper (this call uses the
 ## modified experiment with fewer genes)
 ## anything from files)
 result <- fspma.wrapper(RG=RG, info=ret$info)
## End(Not run)
```

---

fspmaRG.2.RG *Conversion of FSPMA objects to and from RG*

Description

These functions act as adaptors from FSPMA objects to (YASMA's) RG object. They are meant to be used in cases where YASMA functions that operate on RG structures should collaborate with FSPMA.

Usage

```
fspmaRG.2.RG(fspmaobj)
RG.2.fspmaRG(RG, fspmaobj)
```

Arguments

`fspmaobj`	is either a FSPMA object as generated by fspma.wrapper or a FSPMA RG object
`RG`	is a YASMA RG object (and possibly downwards compatible to SMA).

Value

`fspmaRG.2.RG` takes a FSPMA object and returns a YASMA RG object that can be passed on to YASMA functions. `RG.2.fspmaRG` takes s YASMA RG object and a FSPMA object and returns a modified FSPMA object.

See Also

`tabdel2rg` and `fspma.wrapper`

Examples

```
## Not run:
fspma.obj <- fspma.wrapper('twochannel.def')
yasma.rg <- fspmaRG.2.RG(fspma.obj)
## or alternatively with the same result
yasma.rg <- fspmaRG.2.RG(fspma.obj$RG)
## remove 0.1 quantile of least correlated genes
## or call other YASMA functions!
RG <- rg.remove.quantile(RG, 0.1)
## and merge the modified RG with the FSPMA object.
fspma.new <- RG.2.fspmaRG(RG, fspma.obj)
## to allow subsequent fspma analysis use externally
## modified data. e.g. here do the entire run on a reduced gene set.
fspma.res <- fspma.wrapper(RG=fspma.new$RG, info=fspma.new$info)
## End(Not run)
```

`gen.yasma.eqns` *Convert fspma info structure to model equations and additional modelling information.*

Description

The functions info.2.yasmaeqns and gen.yasma.eqns are used to generate a standard set of model equations of ANOVA models that allow to get ANOVA tables, variance components and rank statistics.

Usage

```
info.2.yasmaeqns(info)
gen.yasma.eqns(terms, randeffs, rank.term=1)
```

Arguments

`info`	Data structure obtained from function `read.defs`.
`terms`	Vector of effect names (model terms without gene effect).
`randeffs`	Boolean Vector of random effects flags.
`rank.term`	Index of the model term that defines the ranking we are interested in. This is typically an interaction term of the gene effect and the first effect in terms like a gene:time, a gene:species or a gene:type interaction and the default index will thus be 1.

Details

Converts info to ANOVA equations and additional rank information necessary to apply YASMA to general experiments (like longitudinal problems). gen.yasma.eqns generates default model equations (Regressors are iterative nestings of effects) and additional information that we need to calculate the ANOVA model and test statistics. The resulting model.info structure is used by functions `do.anova`, `aov.tab.varcomp.2.file` and `contrast.rank.2.file`. The most important part of model.info is model.info$eqn, this is the ANOVA model equation used for analysing the data.

Value

Both functions return a model.info object that is not meant to be manipulated at user level.

See Also

`do.anova`, `aov.tab.varcomp.2.file`, `contrast.rank.2.file` and `get.spk.nrm.eqn`

Examples

```
## Not run:
info <- read.defs('deffile.def')
model.info <- info.2.yasmaeqna(info) ## calls gen.yasma.eqns
## alternatively:
terms <- c('time', 'sample')
rand. terms <- c(FALSE, TRUE)
model.info <- gen.yasma.eqns(terms, rand. terms)
## End(Not run)
```

`read.defs` *Read the fspma definition file and check consistency*

Description

Reads the definition file

Usage

```
read.defs(fname, log.fname=fspma.logfile, init.logfile=T, print.logheader=T)
```

Arguments

`fname`	File name of a definition file that controls the entire analysis.
`log.fname`	Name of log file which defaults to fspma.logfile or 'fspma.log.txt', unless overriddenby the user.
`init.logfile`	Controls whether the logfile is initialised before starting the run. Its value defaults to TRUE.
`print.logheader`	Flag that controls whether the starting time is written to the log file.

Details

The function reads the definition file, stores the information in the internally used info structure, which is checked for consistency. The consitency check is meant to avoid that apparent mistakes get caught before processing starts. In this context read.defs is useful as a stand alone function outside `fspma.wrapper`, since it helps debugging the definition file. In that context it is worth mentioning that all problems with the definition file are reported on the R console and written into the log file. All messages point to those definition file entries that could have led to the problem.

Value

Function `read.defs` returns the FSPMA info object. This is an R object that represents microarray data analysis as described in the definition file. Any manipulations at code level should be done with care and bearing in mind that the analysis result is no longer in agreement with the definition file.

See Also

`fspma.wrapper`, `deffile.def`

Examples

```
## Not run:
 info <- read.defs('mouse_testis_adult_day5.def')

## End(Not run)
```

`specify.experiments` *Explains how to set up the definition file for different experimental conditions.*

Description

The package fspma can be used to analyse all experiments, as long as they are reference designs. We may work with double and single channel data, with and without background. The input to the fspma scripts may also be normalized "meta" information.

Usage

```
(1) Genepix spotted files
(2) Bluefuse spotted files
(3) Affymetrix files
(4) Pre-processed (normalized) meta information
```

Arguments

(1)	Specify R, Rb, G and Gb columns (`{R,Rb,G,Gb}.colnam:`). Specify normalization by setting `Normalization:` and finally set `DoRG2logarray: T` (i.e. move to log scale).
(2)	Specify R and G columns (`{R,G}.colnam:`). The channel background is initialized by 0. Specify normalization by setting `Normalization:` and finally set `DoRG2logarray: T` (i.e. move to log scale).
(3)	Specify the R column only (`R.colnam:`). G is initialized with 1 and all channel background with 0. Specify normalization by setting `Normalization:` (**NO** loess normalization without spikes for single channel data!) and finally set `DoRG2logarray: T` (i.e. move to log scale).
(4)	For meta information available as ratios or expression values: Specify the corresponding column as R channel, turn off normalization by setting `Normalization: NA` and set `DoRG2logarray: T`. For meta information available as log expression or log ratio: same as above however with `DoRG2logarray: F`.

Details

In addition one can use spike based normalization and imputing if this is necesary.

See Also

`fspma.wrapper`, `tabdel2rg`, `deffile.def`, `treat.na` and `spike.normalize`.

`spike.normalize` *Normalization of array data with respect to spike probes*

Description

Functions for spike based normalisation.

Usage

```
get.spk.nrm.eqn(info, model.info)
spike.normalize(RG, info, model.info)
```

Arguments

`RG`	Modified (but compatible) yasma RG structure. Contains in addition to the usual entries the slide and grid number.
`info`	fspma's info structure, which has a new spike-gene list.
`model.info`	fspma's model.info structure, which has now a new model equation for spike normalization.

Details

The function spike.normalize normalises array data (in RG format) based on spike's that should be randomly positioned across the array. The normalisation procedure is controlled via the definition file entry Normalisation:, which in this case has to be ”spike” with various options: ”location” removes the within slide location; ”spatial” removes a within slide loess fit conditional on spot position; ”amplitude” removes a within slide loess fit conditional on amplitude. ”spatial” and ”amplitude” can be used together in which case the conditioning is on the space - amplitude interaction. Grid number can be included as interaction factor by specifying the modifier ”grid”. Finally one can remove scale as well by specifying ”scale”. The spike genes are defined together with amplitude factors, spike.name<TAB>R.spike<TAB>G.spike, where only the correct ratio is necessary. Spike based normalisation operates on the residual in log ratio, respectively ”M” space (or on log expression residuals, if the input is one channel arrays). We get the following spike models that are used for a loess fit:

spike	spatial				–>	M-log(R.spike)+log(G.spike)	X:Y
spike	amplitude				–>	M-log(R.spike)+log(G.spike)	A
spike	spatial	amplitude	grid	scale	–>	M-log(R.spike)+log(G.spike)	X:Y:A:grid

in addition the last also divides by std. deviation over spike residuals, estimated for every sub grid separately.

Value

get.spk.nrm.eqn(info, model.info) generates model equations for spike based normalisation and augments model.info. The object model.info is not meant to be manipulated at user level. spike.normalize(RG, info, model.info) applies spike based normalisation to the aray data stored in RG, which it returns.

See Also

`fspma.wrapper`, `read.defs`, `deffile.def`, `info.2.yasmaeqns`, and `treat.na`

Examples

```
## Not run:
## read definition file that specifies spike normalisation
info <- read.defs('mouse_testis_spike_norm.def')
## get relevant modelling information
model.info <- info.2.model.eqns(info)
## add spike model equation to model.info # (normally hidden in fspma.wrapper)
model.info <- get.spk.nrm.eqn(info, model.info)
## and call normalisation
RG <- spike.normalize(RG, info, model.info)
## End(Not run)
```

`tabdel2rg` *Loads the array data according to the specification in the definition file.*

Description

The function will load data from arbitrary tab delimited files, as long as the layout is identical. It can read double colour arrays with and without background information, single colour arrays and pre normalized meta data. See `specify.experiments` for detailed suggestions how to adapt the definition file to various input scenarios.

Usage

```
tabdel2rg(info, log.fname=fspma.logfile)
```

Arguments

`info`	control structure as obtained by `read.defs`.
`log.fname`	name of logfile, defaults to fspma.logfile, which is 'fsmpa.log.txt' unless reset by the user.

Details

Loading arrays is controlled by all column entries in the info structure (column names). The flag info$load.fast, which contains the value of the "load.fast:" entry in the definition file allows to speed up data loading if and only if all files have the genes on the same positions. The list info$dat.file.rd, which contains all entries of the "file.alloc:" description in the definition file allocates files to a corrsponding set of levels. Each entry of this allocation consists of a file name and its allocation. The final flag is a colour swap indicator.

`R35_NIA1_AWX_ad_612_Fl_output.xls 1 1 F` specifies that this file represents data that corresponds to the first level in the first effect (here time) and to the first level in the second effect (here a technical replication). The last column specifies that this array is not a colour swap.

The function first checks whether the column header information corresponds to the predefined setting in the definition file and then reads gene names and the array data (including a spotter flag that indicates invalid spotter results). After all data has been read, the function builds an RG structure.

Value

tabdel2rg reads microarray data from tab delimited files and returns an RG object. The information in RG is downwards compatible with YASMA's RG. It is however extended, to take bad quality markers and spikes into account.

See Also

`fspma.wrapper`, `read.defs` and `deffile.def`.

Examples

```
## Not run:
## read definition file
info <- read.defs('deffile.def')
## read data
RG <- tabdel2rg(info)
## End(Not run)
```

`treat.na` *Treat flaged (missing) Spots and NA values*

Description

This set of functions is provided to resolve NA values in the RG structure that may arise because they were contained in the original data. NA is also set in case that the flag information for spots is loaded and a flag equals the info$flag.nack entry (Flag.NOK: line in the definition file). The behaviour in the context of missing values can be controlled by the "Impute.Mthd:" entry in the defnition file. Options are knn<TAB>n, del or NA. The latter implies no action. After issuing a warning message, "no action" is however overridden by del, if NA's are found in the data. This user request was added, to avoid that processing terminates unsucessfully if NA's are left untreated.

Usage

```
treat.na(RG, method, go2log=T)
knn.impute(RG, k=10, go2log=T)
del.impute(RG)
```

Arguments

`RG`	A YASMA RG object, who's NA entries will be modified by k nearest neighbour imputation or deleted.
`method`	A method string that is 'knn' for k nearest neighbour imputation (Troyanskaya et.al.), 'del' if NA entries are resolved by removing the corresponding genes and 'NA' if there is no action taken.
`go2log`	Optional flag, specifying whether data should be moved to log scale before imputing.
`k`	Number of neighbours used for calculating the knn imputation.

Details

These functions provide possibilities to resolve missing spot issues or NA values in RG colour channels. The original proposal in yasma to use the ANOVA model for imputing fails to work due to the large memory requirements of the corresponding linear model.

Value

These functions return RG objects, with bad quality spots corrected or the corresponding genes removed.

See Also

`tabdel2rg`, `deffile.def`, `fspma.wrapper`

Examples

```
## Not run:
## read the data
RG <- tabdel2rg(info)
## set the channel entries of missing or failed spots to NA:
if(!any(is.na(info$flag.nack)))
  {
    set.na.dx <- RG$Flg==info$flag.nack
    RG$R[set.na.dx] <- NA
    RG$G[set.na.dx] <- NA
  }
## now we treat NA according to info$impute.mthd
RG <- treat.na(RG, info$impute.mthd)
## alternatively we can do this by hand and call
RG <- knn.impute(RG, 20) ## using the 20 closest profiles to impute
## or remove the corresponding genes
RG <- del.impute(RG)
## End(Not run)
```

Acknowledgements

The development of the FSPMA library was funded by the BBSRC's Exploiting Genomics initiative under ref. 8/EGH16106, "Shared Genetic Pathways in Cell Number Control".

Bibliography

[1] F. Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In W. Härdle and B. Ränz, editors, *Compstat 2002 ÂŮ Proceedings in Computational Statistics*, pages 575–580, Heidelberg, Germany, 2002. Physika Verlag. URL [http://www.ci.tuwien.ac.at/ leisch/Sweave].

[2] P. Sykacek, R. Furlong, and G. Micklem. A Friendly Statistics Package for Microarray Analysis. Technical report, Departments of Pathology & Genetics, University of Cambridge, 2005. [Available at `http://www.ccbi.cam.ac.uk/software/psyk/software.html#sykacek_etal_TR051`].

[3] G. O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[4] L. Wernisch, S. L. Kendall, S. Soneji, A. Wietzorrek, T. Parish, J. Hinds, P. G. Butcher, and N. G. Stoker. Analysis of whole-genome microarray replicates using mixed models. *Bioinformatics*, 19(1):53–61, 2003.

[5] Snedecor, G. W. (1934) *Calculation and Interpretation of Analysis of Variance and Covariance.* Collegiate Press, Ames, Ia.