A short reference to FSPMA definition files

P. Sykacek
Department of Genetics &
Department of Pathology
University of Cambridge
peter@sykacek.net

June 22, 2005

Abstract

This report provides a brief reference of FSPMA's definition files. The code of the library is closely linked with code from YASMA and thus provided under the GPL 2 license. The most important implication of the GPL 2 license is that FSPMA is free software and comes with NO WARRANTY.

1 Functionality

FSPMA, [3], is an R-library that is useful to analyse microarray data. FSPMA's concept is based on a definition file that describes the experiment one would like to analyse and the way how that should be done. The definition file allows analysis without adapting or writing R-scripts and serves as documentation. The library applies to data from different platforms (single and two colour arrays with optional preprocessing steps done before the data gets loaded by FSPMA). The main restriction of FSPMA is that the experiment must be a balanced reference design. Analysis includes handling of bad quality flagged samples, normalization that allows to use spike RNA, calculation of an ANOVA table and variance components and finally gene ranking based on within ANOVA contrasts and by using an ANOVA model per gene.

2 Installation

FSPMA can be downloaded from `http://www.ccbi.cam.ac.uk/software/psyk/software.html#fspma`. The library is wrapped around YASMA, [5], which is available at `http://people.cryst.bbk.ac.uk/wernisch/yasma.html`. For the users convenience, we provide a slightly modified version of YASMA for download as well. This version has some modifications that were necessary for compatibility with R 2.1 and to allow to provide with a win32 port. Both packages can be installed by downloading the apropriate zip archive from FSPMA's web page. The archive must be unzipped into a local directory and then one calls the apropriate shell script to install the library (fspmawininstall.bat under Windows and fspmaxinstall.sh everywhere else). The library comes with a set of five short experiments that allow the user to experiment with definition files in situations where data is available as quantified two colour arrays, single channel data and processed single channel data (e.g. obtained using some preprocessing tool). More information about these examples is found in the R help pages that accompany this library. The definition file discussed below as `http://www.ccbi.cam.ac.uk/software/psyk/srcfiles/tstsgd_A.def.gz` and provided at FSPMA's homepage. Allthough it is not required to understand this document, one migh want to run this analysis. In that case it is necessary to obtain the corresponding Affymetrix arrays from the NCBI GEO Datasets at `http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?CMD=search&DB=gds` under reference GDS660 [2].

This tutorial will discuss all entries in this definition file and provide an example how Affymetrix data can be analysed. Having copied the data and the definition file (unzipped) into a local drectory, analysis requires to start R in that directory and just type the commands

```
>> library(fspma)
>> ret< −fspma.wrapper(’tstsgd_A.def’)
```

at the R command line. In case of success, this analysis will produce several (tab delimited) files that contain the ANOVA table, test results, the normalized and reformatted raw data and a corresponding level description of the experiment. The latter two files are useful as inputs to further analysis. Using those files, one does not have to worry about inconsistent file layouts. FSPMA based analysis can entirely be controlled via the definition file. During analysis, this file is parsed and the relevant data structures generated. These are in turn used to load data, normalize it, interface to YASMA to get the ANOVA table and variance components and to generate gene lists for all tests specified in the file.

3 Description of the experiment

Lines in the definition file starting with # are comments and ignored. All other entries have to separate values from keys and from each other by *one* tabulator. YASMA is a mixed model ANOVA tool, which requires to specify model equations and information whether effects are random or fixed. The corresponding part of the definition files are the following lines:

```
# 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:        5         2         1
# 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
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:     t11.5dpc t12.5dpc t14.5dpc t16.5dpc t8.5dpc
```

This experiment consists of 5 time points with two samples at each. Although there are no replicate genes within a file, we need to specify the number of replicates, which is thus 1. Both sample and replicate are random effects, since for those we can not assume that the experiment contains an exhaustive list of all possible levels. Time point on the other hand is a fixed effect, because the analysis does not aim to generalize from the time points in the experiment to others. Rank.Eff allows to specify over which effect we want to perform gene ranking. Here this is time and we have 5 levels we name as is shown under Rank.Names. Although these names are arbitrary, they are later used as R variable names and should thus be chosen accordingly. **Is.Unbalanced:** allows to load and normalize unbalanced experiments (i.e. data, where e.g. the number of samples varies with time point.). Analysis of such experiments is not possible. This is prohibited by the huge dimensionality one would get with conventional ANOVA methods. YASMA's uses an efficient ANOVA implementation which requires balanced designs but does not have this problem.

4 List of genes

```
# A list of all gene names (as found in gene.col in the data).
# This list is the final part of the definition file. This is necessary to allow to cope
# with experiment designs, where different genes come in different numbers of replicates
# on a slide. An experimental protocol that does not make any sense, since we must
# remove these genes randomly if we do not want to bias the analysis towards those with
# multiple replicates. It is much more sensible to use this "space" for spike genes.
# Below we have the Affymetrix U74 mouse genome id's.
genes.list:
100001_at
100002_at
100003_at
100004_at
```

```
100005_at
100006_at
100007_at
100009_r_at
100010_at
100011_at
100012_at
100013_at
100014_at
100015_at
100016_at
# and many more...
```

The gene list is found under the key **gene.list:**. Since this is a quite lengthy list, it is recommended to put that at the end of the definition file. During data loading all replicates of the genes from this list are searched in the **gene.colnam:** column of each data file. All genes that are in excess from what is specified as (minimal) replicates on slide (the last value in the **Effects:** definition) are discarded. This has no effect here, since these files (Affymetrix data) contain exactly one entry per gene. It is, independent of our requirement of having a balanced design, in general not a good idea to replicate some genes more often than others. These additional replicates must be discarded at random since every other use is prone to bias the analysis results in favour of those genes that have larger numbers of replicates. A much more sensible use of that space is to fill it with targets for spike RNA of known concentration that can be used to normalize the data.

5 File allocation

Analysis requires next to assign the data of individual files to the corresponding levels of the experiment. Hence in this example we need to specify which file contains the data of each time point and sample. This is done via a list that starts with the key **file.alloc:**. Note that here the values are taken until the parser reaches a different key element or the end of the definition file. Flag **fast.load:** allows to speed up analysis considerably (here using **fast.load: T** reduces runtime to a quater). FSPMA searches for genes in the slide files. If we know that all files have identical structure, this is necessary only once and thus reduces the ammount of time consuming searching. However, if not all files have the same structure, using **fast.load: T** will produce meaningless results. If there is any doubt, do not trade valid results for speed!

```
# Fast Load? We know that this data is homogeneous.
# hence there is no doubt about identcal gene positions.
fast.load:     T
file.alloc:
#MGU74A        time   sample   color swap?
# 11.5 dpc
GSM21973.txt   1      1        F
GSM21976.txt   1      2        F
# 12.5 dpc
GSM21979.txt   2      1        F
GSM21982.txt   2      2        F
# 14.5 dpc
GSM21985.txt   3      1        F
GSM21988.txt   3      2        F
# 16.5 dpc
GSM21991.txt   4      1        F
GSM21994.txt   4      2        F
# 18.5 dpc
GSM21997.txt   5      1        F
GSM22000.txt   5      2        F
```

In addition to the file name and the level of the effect (time point and sample within time point) one specifies a flag that allows to load dye swap data. These entries are either 'T' for 'true' or 'F' for 'false'. As we analyse Affymetrix data, they must all be false. This is however the *users responsibility*, since the data could equally well be pre-processed log ratios of dual channel expressions with dye swaps not taken care of!

6 Localising data and different data types

Within files the data can be in arbitrary columns. To avoid failure in cases, where the file structure changes within an experiment, FSPMA identifies columns by their names. This is done by the following set of keys.

```
# gene column name in the header
gene.colnam:       ID_REF
# Cy5 spot column name in the header
R.colnam:          VALUE
# 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
# Cy3 background column name in the header (NA if none)
Gb.colnam:         NA
# flag column name in the header (NA if none)
Flag.colnam:       NA
# X-pos column name (NA if none)
X.colnam:          NA
# Y-pos column name (NA if none)
Y.colnam:          NA
# Grid.no. column name (NA if none)
Grid.colnam:       NA
```

Except for the columns that contain the gene identifiers and the R-color channel, all other columns may be missing. This is indicated by putting the keyword NA into the corresponding row. Since it is closely related, it is at that point also advisable to look at the **DoRG2logarray:** entry, which specifies whether data is put onto log scale or taken as is.

```
# control conversion RG − > array (T − > take log, F − > use as is)
DoRG2logarray:   T
```

This set of keys allows that FSPMA can be used to analyse all experiments, as long as they are balanced 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. In particular we have the following options:

```
1) Genepix spotted files (two color with background)
2) Bluefuse spotted files (two color no background)
3) Affymetrix files (single channel arrays)
4) Pre-processed (normalized) meta information
```

These different cases require the following setting:

1) Specify R, Rb, G and Gb columns (**{R,Rb,G,Gb}.colnam:**). Specify normalization by setting **Normalization:** and finally set **DoRG2logarray: T**.

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**.

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**.

4) For normalized 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**.

7 Bad quality flags and Imputing missing values

```
# Value for flagged entries. If available checked against Flag. column NA = absent
Flag.NOK:        NA
# How do we impute: (NA for none, knn <TAB> k for knn using k neighbours
# and del for removing such genes)
# Impute.Mthd:   knn     20
# Impute.Mthd:   del
Impute.Mthd:     NA
```

FSPMA regards the information read from **Flag.colnam:** column as indicator of bad quality flagged spots if this value corresponds to one specified under the **Flag.NOK:** key. After having read the data expression values are set to NA in all marked rows. After that all NA expression values have to be taken care of. Depending on the value under key **Impute.Mthd:**. If the value of that key is **del** this is done by removing the corresponding genes from the analysis (i.e. If a single expression value of a gene is flagged, the gene is removed from the analysis.) The second option is **knn n** or “k nearest neighbour imputing” that was suggested by [4]. No imputing is selected by having NA as method. If NA’s are found in the data, FSPMA issues a warning and overrides this choice with **del**. This is a user request since otherwise the analysis will fail.

8 Normalization

There are two types of normalization available. Classical approaches that are handled by YASMA and spike based normalization, which is described in the next section.

```
# normalization control NA if none, otherwise loess,
# location or scale combinations like loess scale or
# location scale are allowed as well.
# e.g. no normalization:
# Normalization: NA
# e.g. remove mean:
# Normalization: location
# e.g. remove a loess fit of an M/A representation (default span 0.9 degree 1)
# Normalization: loess
# e.g. loess with specification of span and degree
# Normalization: loess    0.7     1
# e.g. remove mean and convert to unit std. deviation
# Normalization: location scale
Normalization:   NA
```

In the context of analysis of single channel data **loess** is an invalid option which results in an error message. This is necessary since loess based normalization operates on the M/A (i.e. log ratio log amplitude) representation. For single channel data this is a line that will be fit perfectly by a loess fit. As a result all expression values will up to numerical inaccuracies be zero. Value **location** in that key will remove the mean log ration of each slide. Value **scale** transforms the data to a common dynamic range (i.e. changes within slide variance).

9 Spike based normalization

```
# FSPMA 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 spot
# position, grid number and amplitude are possible. The latter fits a loess
# fit to the spikes residuals and subtracts that from all other gene values.
# 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. 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. After calculating the overall
# scale, each slide is in addition transformed to that scale.
Normalization:   spike     0.95      1         spatial   amplitude grid     scale
# In addition one has to provide a spike list with format
# name             R_concentration   G_concentration. R and G refer to the
# above channel names and are the known spike concentrations in that
# channel (only the correct ratio is important). spike.list:
# spike name       R_con.    G_con.
spike_01_A01       1         1
spike_01_A02       2         1
spike_01_A03       3         1
spike_01_A04       2         2
spike_01_A05       1         2
spike_01_A06       1         3
spike_01_A07       3         3
#... and more if available
```

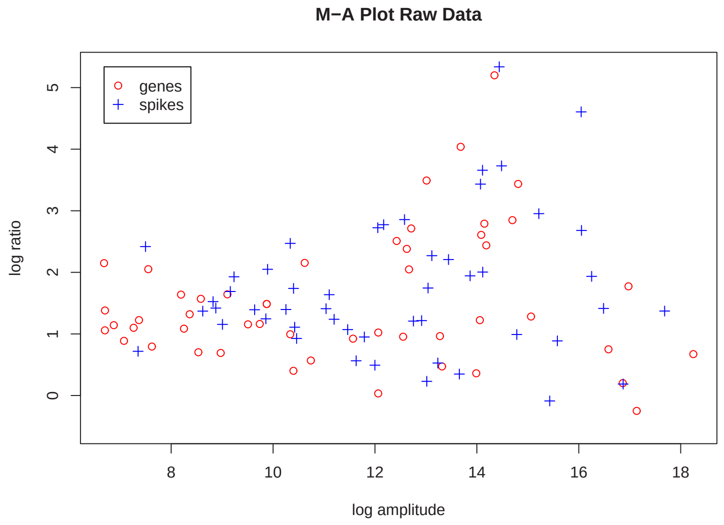

Figure 1: Scatter plot of log ratio over log amplitude (M/A-plot) of genes and spike RNA. The data gives the impression that there is a tendency towards up-regulation (positive log ratios). Since this also happens for the spikes, which are known to have identical concentration in both channels, this is due to a technical bias.

In addition, FSPMA allows spike based normalization. This is selected using the value **spike** under the **Normalization:** key. The function spike.normalize normalizes array data (in RG format) based on spike's that should be randomly positioned across the array. The normalization procedure is controlled via the definition file entry **Normalization:**, 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 R.spike G.spike**, where only *correct ratios matter*. Spike based normalization 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 in loess fits:

spike spatial $\rightarrow M - \log(R.spike) + \log(G.spike) \sim X : Y$
spike amplitude $\rightarrow M - \log(R.spike) + \log(G.spike) \sim A$
spike spatial amplitude grid scale $\rightarrow M - \log(R.spike) + \log(G.spike) \sim X : Y : A : grid$

In addition to the shown loess equation, the last example also rescales the data based on the spike residual log ratios to the experiment wide std. deviation.

We illustrate spike based normalisation with a preliminary calibration experiment of some spike RNA on a two channel platform. Figure 1 shows the M-A representation of spikes and some genes before normalisation. The theoretical spike concentration is identical for all spikes. Since the measured log ratio is positive for genes and spikes, we have to conclude that there is a positive bias in the measurement process. Normalising this data with an amplitude based loess fit towards the residual log ratio with additional scale removal (**Normalization: spike amplitude scale**), we obtain the situation displayed in figure 2. It is clear to see that the normalized data shows less bias. In contrast to the non-normalised case, we can now see that genes at larger spot amplitude are both up and down regulated.

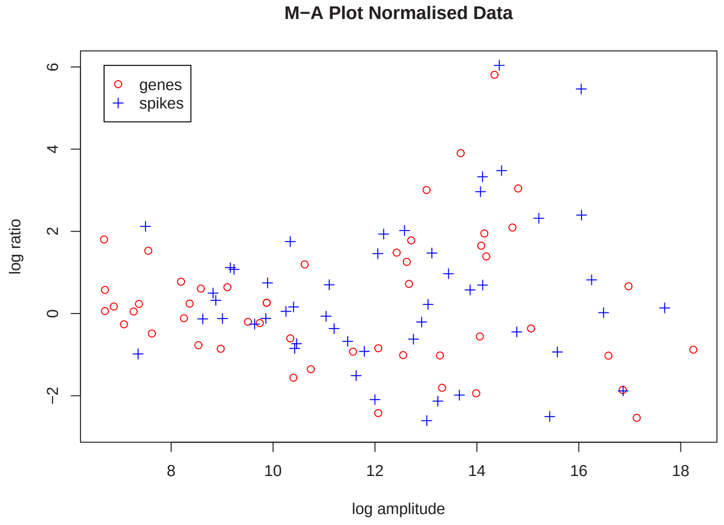

Figure 2: Scatter plot of log ratio over log amplitude (M/A-plot) of genes and spike RNA after normalisation. After normalisation, the scatter plot of the same data shows less bias. It is now apparent that at larger amplitudes, we find both up and down-regulated genes.

10 ANOVA table and variance components

```
# ANOVA table and variance component output file
ANOVA.Outfile: r1_tstsgd_A_aov.txt
```

The only option one has to specify is which file should be used to store the ANOVA table and the variance components. YASMA's ANOVA functions are used by providing the data, ANOVA model equations and Boolean indicators about which of the effects are random. In order to automate analysis, FSPMA converts

Table 1: Top five genes, ranked according to contrast “t13.v.t45”

dx	g.nams	p.val		11.5d	12.5d	14.5d	16.5d	18.5d
2732	103535_at	1.68E-100	↓	2.92	3.04	2.75	-0.02	-0.48
10407	97180_f_at	1.79E-76	↓	3.05	3.03	2.91	0.59	-0.08
372	100549_at	2.68E-59	↓	2.79	2.56	1.79	0.08	-0.13
11792	99058_at	8.23E-55	↓	2.20	2.02	1.42	-0.14	-0.72
1917	102416_at	4.26E-43	↑	-0.86	-0.62	2.09	2.41	2.08

descriptions of experiments as were discussed in section 3 into a default ANOVA equation. This equation is constructed in an iterative manner by adding each effect as interaction term to the grouping of the previous level. The experimental description in section 3 leads thus to the ANOVA equation

$$M \sim G + G : time, \tag{1}$$

where M denotes log expression (or log ratio), G is the gene index and $time$ represents time points. “Sample” is not part of the model equation, because that gave as many parameters, as we have data points. The model would explain the data perfectly and residuals were zero. In order that inference makes sense, we must allow that the samples estimate the residual variance. Representing M as response y, G as index g, $time$ as index t and using n as sample index, this ANOVA corresponds to the linear model

$$y_{g,t,n} = \mu + \alpha_g + \beta_{g,t} + \epsilon_{g,t,n}, \tag{2}$$

where μ is a global mean, $\mu + \alpha_g$ the expected expression for gene g and $\mu + \alpha_g + \beta_{g,t}$ the expectation for gene g at time t. The $\epsilon_{g,t,n}$ are assumed to be i.i.d. Gaussian distributed and represent the residuals in the model. Note that the variance of that Gaussian represents the variance associated with the random effect “sample”. The number of on slide replicates is one and has thus no effect on the ANOVA model. The resulting ANOVA tables and variance components are obtained by calling the appropriate YASMA functions. For the experiment defined in section 3, we get the following output.

ANOVA Table:

effects	df	SS	MS	F	p
G	12421	1.164e+05	9.3711	178.74	<1.0e-15
time,Gtime	49688	4555	0.091671	1.7485	<1.0e-15
resids	62110	3256.4	0.05243		

p: 62109 n: 124220 R^2: 0.973783 R^2(adj): 0.9475675
Sp: 9768.996 Cp: 3256.402 AIC: -2.641447 Schwarz IC: 2.223379 SHQ IC: -1.179338

After having obtained very significant p-values which support to reject the null hypothesis, that all gene-time interactions have zero $\beta_{g,t}$, it makes sense to further analyse which of those are non zero.

11 Gene Ranking

```
# The following lines specify gene ranking.
# Example for ranking based on “average log differential expression”
# Invalid for single channel data but valid, if a single column in the input file
# contains log ratios.
# Base.Contrast: VARIETY
# Example for a contrast comparing time point 5 versus time point 1
# Base.Contrast: ct15     1     0     0     0     -1
# Example for general ranking against first time point
# blone : base level one (expands in this context implicitly to 4 pair-wise contrasts).
# Base.Contrast: blone    1
# additional contrasts such that all possible pair-wise comparisons are done:
# Base.Contrast: d12.5d14.5     0     1     -1     0     0
# Base.Contrast: d12.5d16.5        0        1        0        -1        0
# Base.Contrast: d12.5d18.5        0        1        0        0         -1
# Base.Contrast: d14.5d16.5        0        0        1        -1        0
# Base.Contrast: d14.5d18.5        0        0        1        0         -1
# Base.Contrast: d16.5d18.5        0        0        0        1         -1
# Example for a contrast that tests for significant differences
# between the average expression in the first three time points and
# the last two.
Base.Contrast:     t13.v.t45-1      -1       -1       1        1
# ANOVA based ranking : for each gene we calculate an ANOVA and test
# whether all time points share a common mean.
Base.Contrast:     ANOVA
# Final part of the output file names to store the ranked gene lists.
# Use NA if not required. The extension .tsv is for tab separated
# files that load conveniently into MS Excel. Each comparison generates
# a separate file. The first part is always constructed from the name
# given to the comparison. In addition a base level specification will use the
# specified level names.
Contrast.Outfile:  r1_tstsgd_A.tsv
# Type of adjustment: ”holm”,”hochberg”,”hommel”,”bonferroni”,”fdr”,”none”
Comp.Type:         fdr
# p-value threshold or integer number (used as top counter):
Comp.Thrs:         0.05
```

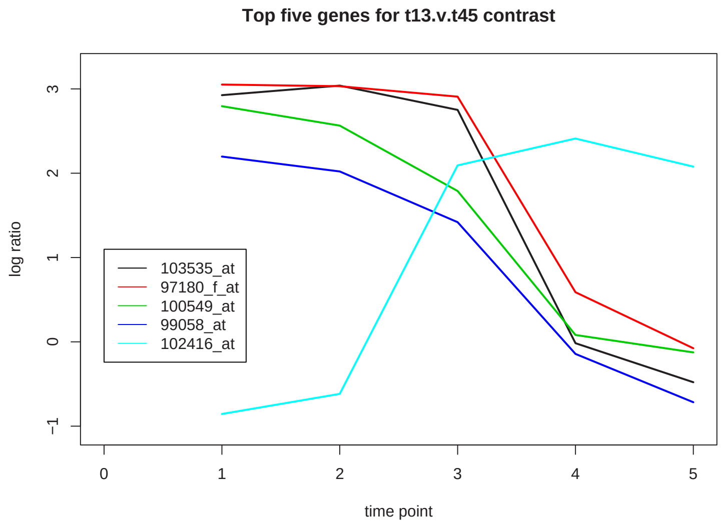

Figure 3: Normalized log ratios of the top five genes, when ranked according to contrast ’t13.v.t45’.

FSPMA provides three options for ranking genes.

- Ranking by average differential expression of each gene. This option is only suitable for two colour arrays (or metadata which are log ratios). As this is the method available in YASMA, FSPMA only wraps around the respective functions there.
- ANOVA based ranking, where a gene based ANOVA is used to obtain the p-value of an F-statistic. This statistic tests the null hypothesis that all “within gene” groupings of the rank effect have identical mean.
- Contrast based ranking, where arbitrary contrasts in the rank effect can be used to specify a null hypothesis. A special case are pair-wise tests, where FSPMA allows as a short cut to specify a “base level”. This specification expands in several pair-wise tests, where each remaining level is tested against the base level.

The latter two options extend YASMA and are useful to analyse more general experimental settings like longitudinal studies. The definition file allows to specify these rank options as well as methods to adjust p-values for multiple comparisons. To avoid large numbers of false rejections, it is vital to put all tests into one definition file. FSPMA will then count the overall number of tests an use that quantity for adjusting the p-values.

FSPMA produces for every comparison a separate tab delimited results file. The columns in that file are, the gene index in the raw data output, the gene identification, the p-value and for all tests except ANOVA a Boolean information whether the gene is up-regulated. Up-regulated means that the average expression after summation over the contrast is positive. The final columns contain the average expression at each level of the rank effect. Here that would be each time point.

The result of ranking genes based on significance levels obtained with contrast 't13.v.t45' is stored in the file "test.t13.v.t45r1_tstsgd_A.tsv". The five genes ranked top according to this contrast are shown in table 1 and a visual impression of the average of the normalized log expression values of the same genes is illustrated in figure 3.

A similar table is stored in "anova.rank.r1_tstsgd_A.tsv" which contains the gene list obtained by ANOVA ranking. The average expression values of those five genes that were ranked top by this test is illustrated in figure 4.

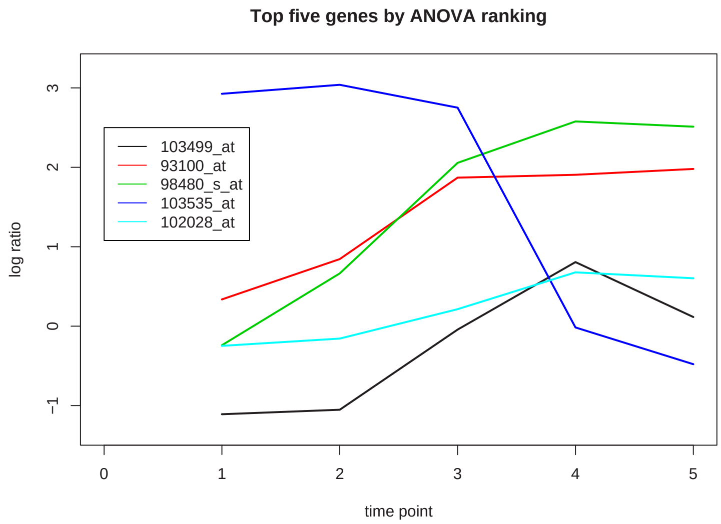

Figure 4: Normalized log ratios of the top five genes, when ranked according to the p-value of a gene specific ANOVA model.

12 What next?

After having gone through this tutorial it is advisable to explore different options provided by the library. This is best done using a dataset that is of interest and by consulting the online help pages provided with FSPMA.

Acknowledgements

This work was funded by the BBSRC's Exploiting Genomics initiative under ref. 8/EGH16106, "Shared Genetic Pathways in Cell Number Control".

References

[1] S. Dudoit, Y. H. Yang, M. J. Callow, and T. P. Speed. Statistical methods for identifying differentially expressed genes in replicated cdna microarray experiments. Technical report, Dept. of Statistics, University of Berkley, 2000. Available at `[http://www.stat.berkeley.edu/users/terry/zarray/Html/papersindex.html]`.

[2] C.L. Small, J. E. Shima, M. Uzumcu, M. K. Skinner, and M. D. Griswold. Profiling gene expression during the differentiation and development of the murine embryonic gonad. *Biol Reprod.*, 72(2):492–501, 2005.

[3] 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`]

[4] 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.

[5] 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.