Using FSPMA

Peter Sykacek[1,2], Rob Furlong[2]
[1]Department of Genetics &
[2]Department of Pathology
University of Cambridge
peter@sykacek.net

June 16, 2005

Abstract

The purpose of this document is to show, how to set up basic definition files for FSPMA and illustrate some further functionality of the library. The data used for the experiments was extracted from a mouse testis development time course. Analysis captures in essence most scenarios one can find in microarray experiments. The examples cover quantified CDNA type data (or other two channel technology), single channel data like Affymetrix and preprocessed meta information. All experiments can be replicated by the user since all relevant data is provided with the library. The code of the library is at several occasions 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 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 directory FSPMA is installed in. It can be found in R using FSPMA's online help and following the link package overview.

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 "fspma.Rnw". After copying all zip archives and the make file into a local directory, the commands "make preprnwrun" and "make fspma.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 purpose. 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.

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 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

Attaching package: 'nlme'


        The following object(s) are masked from package:stats :

         contr.SAS



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

Run started on - Thu Jun 16 2005 13:00:22

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
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,-0.333333333333333
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.

3.1 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.

3.2 Comments on "twochannel.def"

3.2.1 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 onto 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 analysis are not of interest. The rank effect - in this case time - is always a fixed effect. Here we do thus 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 for 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.

3.2.2 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 put 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

3.2.3   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 indicator. 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.

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

3.2.4   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
```

3.2.5   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

3.2.6   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

4   Single channel and preprocessed data

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 - Thu Jun 16 2005 13:00:49


Loading info from file.
Checking consistency of the definition file onechannel_affytype.def was successful.
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
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.
```

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.

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 - Thu Jun 16 2005 13:01:14


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
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 - Thu Jun 16 2005 13:01:38

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
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

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 - Thu Jun 16 2005 13:02:03

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
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.
```

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)
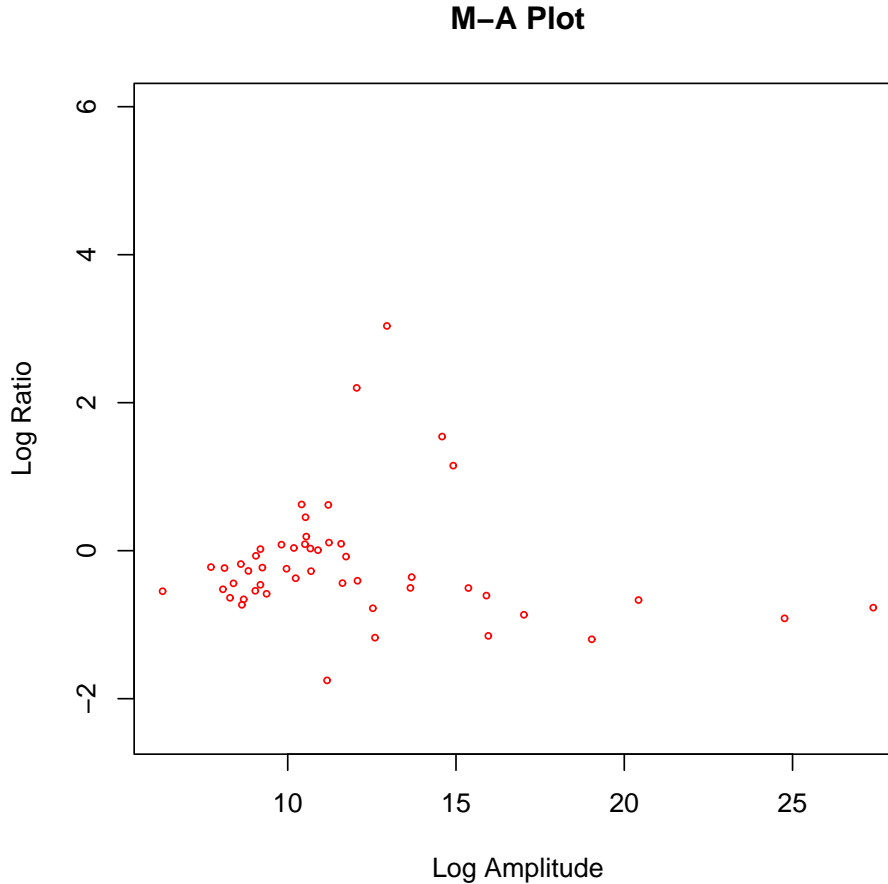
M–A Plot

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.

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")
```

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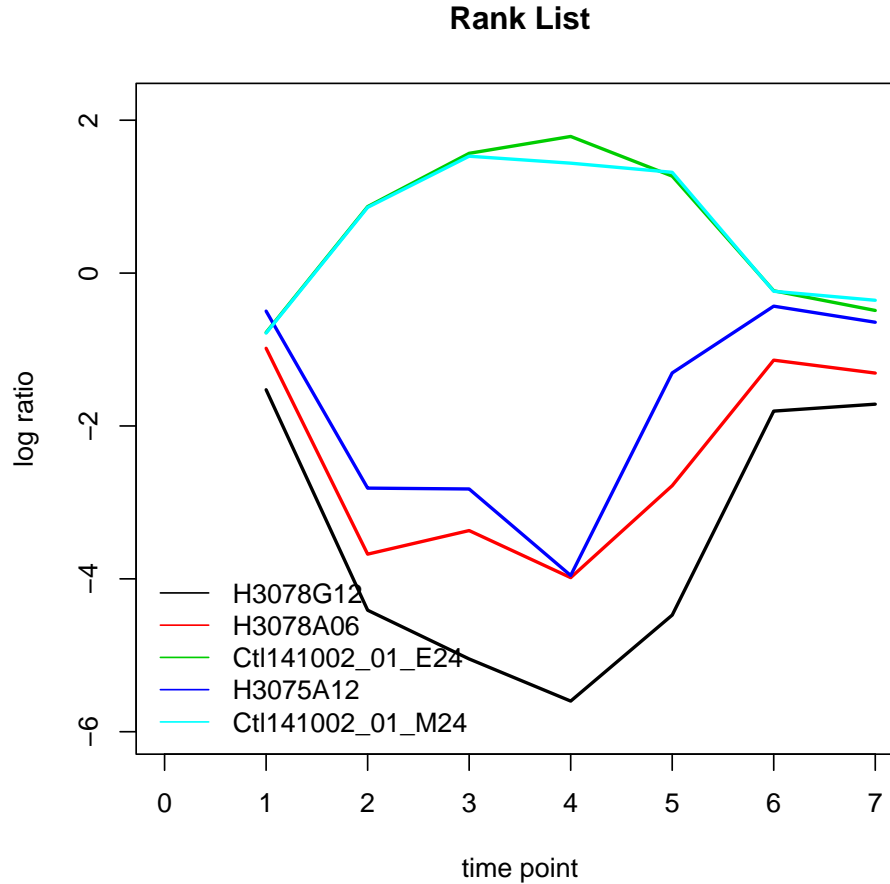
Again there are more options available. Details are found in the online help.

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 advantage to use contrast 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 now be used to produce scatter plots of average channel intensities and illustrate those for a specified number of top up-, down regulated and a 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")
```

## Scatter Plot Average Expression
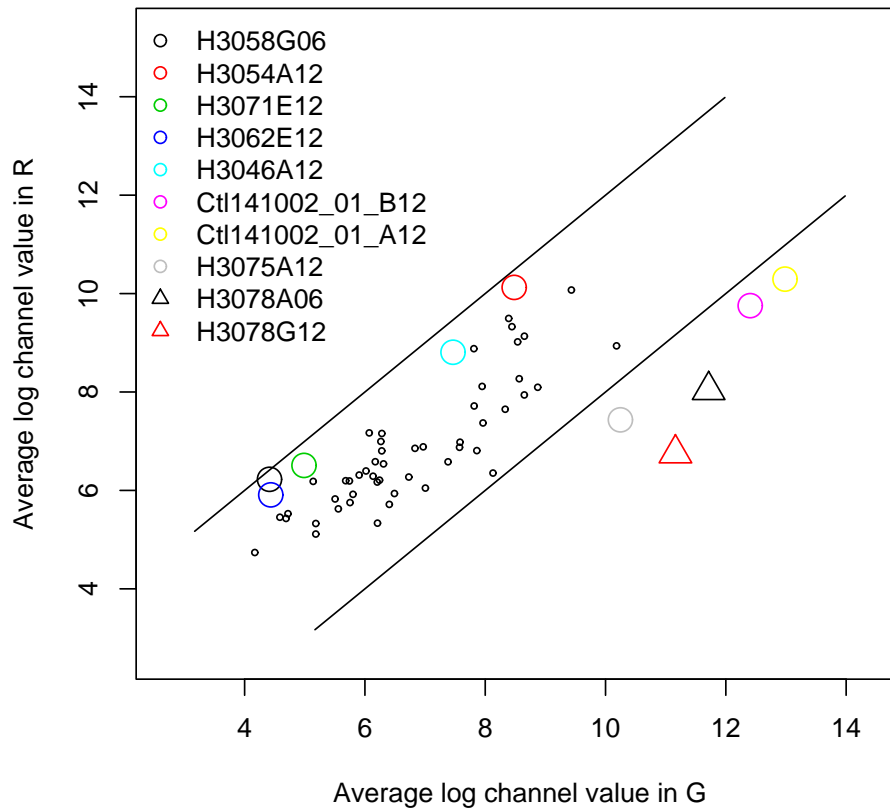
Function **fspma.avchannel** also allows to specify a contrast. In that case we get a plot of log R differences over log G differences. We illustrate this here by plotting the average channel differences of day 1 to day 15 minus day 23 to adult (see also a similar contrast in "twochannel.def"). For plotting 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")
```

Scatter Plot Average Expression
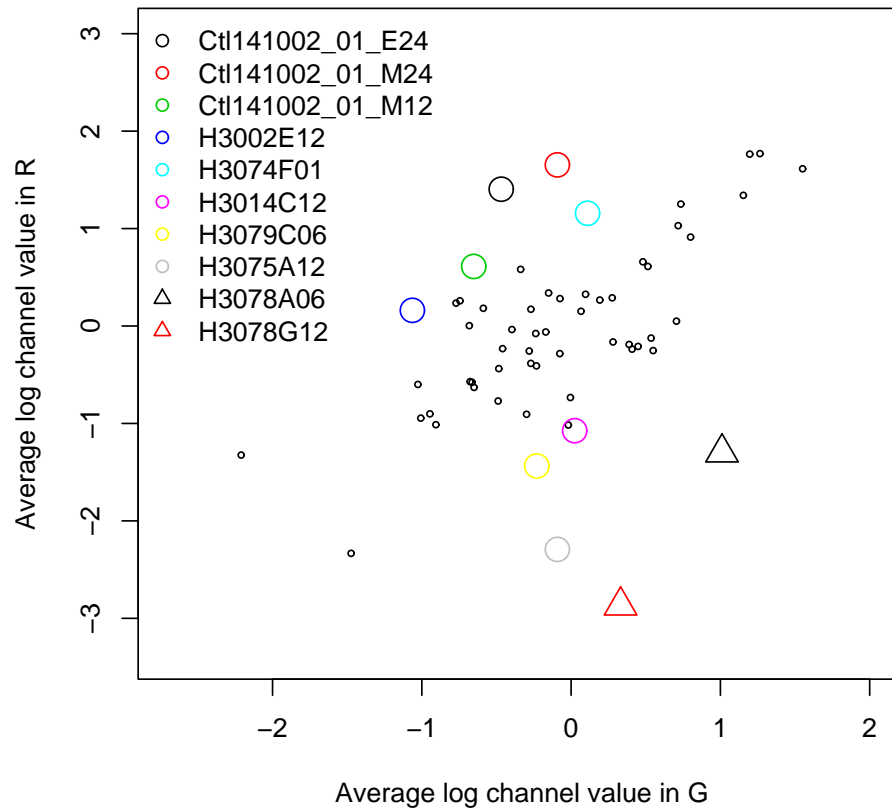
The final option with these functions is to obtain a representation of average log ratio differences between two effects. This representation is again useful for all data sources (i.e. plot $\log2(R[n]/G[n])$ over $\log2(R[m]/G[m])$ for (m,n) being two levels in the rank effect). This is done on the 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 1 over adult",
+     x.leg = "Average log ratios adult", y.leg = "Average log ratios day 1",
+     lg.diff = NULL)
```

Average log ratios day 1 over adult
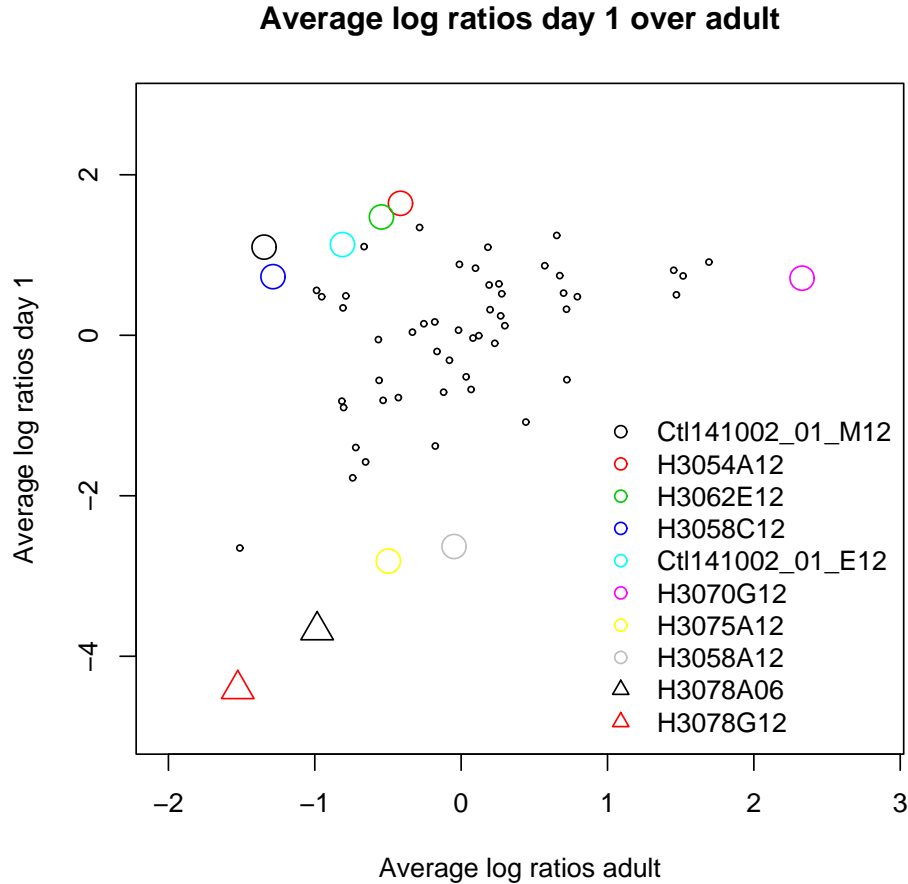
Further options to **fspma.av.scattp** can be found in the online help of the library.

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 - Thu Jun 16 2005 13:02:31

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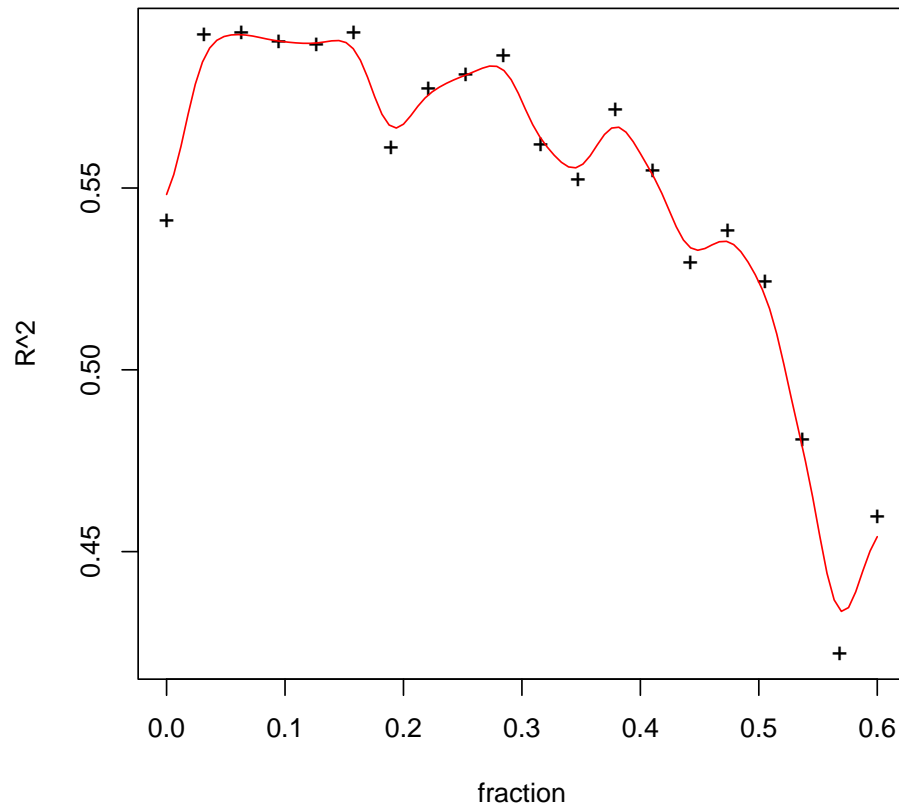
To use the data loaded by **fspma.wrapper** with functions that know about RG objects, we first have to extract a compatible RG object.

```
> 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 - Thu Jun 16 2005 13:03:21
```

```
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
```

```
.....................................................
.....................................................
.....................................................
.....................................................
.....................................................
```

```
..............................................
..............................................
..............................................
..............................................
..............................................
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.
```

7   Next Steps

For more information about 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**. There is also extensive information about how to set up different definition files. In that context we would also like to point to the reference of FSPMA that can be downloaded from our web page at `http://www.ccbi.cam.ac.uk/SOFTWARE/psyk/fspma.html#sykacek_TR051`.

Acknowledgements

References

[1] F. Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In W. Härdle and B. Ränz, editors, *Compstat 2002 Ů 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.sykacek.net/pubs.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.