
`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)
```