Package 'FSAtools'

August 18, 2023

Type Package

Title Fragment Analysis and Capillary Sequencing Tool Kit

Version 2.0.5

Date 2023-08-18

URL https://bioinformatics.ovsa.fr/FSAtools

BugReports https://github.com/maressyl/R.FSAtools/issues

Description A flexible and interfaced framework for importing, processing and ploting Applied Biosystems data files. Application to Reverse-Transcriptase Multiplex Ligation-dependent Probe Amplification (RT-MLPA) gene-expression profiling and classification is illustrated in Mareschal, Ruminy et al (2015) <doi:10.1016/j.jmoldx.2015.01.007>. Gene-fusion detection and Sanger sequencing are illustrated in Mareschal, Palau et al (2021) <doi:10.1182/bloodadvances.2020002517>. Examples are provided for genotyping applications as well.

Note FSAtools replaces and generalizes the former MLPA package from the same authors.

Depends graphics, grDevices, stats, utils, R (>= 2.10)

Imports methods

Suggests Biostrings, parallel, tcltk, tools

License GPL (>= 3)

NeedsCompilation no

Author Sylvain Mareschal [aut, cre],
Philippe Ruminy [dtc, ctb],
Jean R. Lobry [ctb],
Fabrice Jardin [ths]

Maintainer Sylvain Mareschal <mareschal@ovsa.fr>

Repository CRAN

Date/Publication 2023-08-18 18:12:35 UTC

R topics documented:

`add.model` *Object constructor for binary predictors*

Description

This function aggregates the data required to predict class in `classify`.

Usage

```
add.model(x, model, groupMeans, groupSDs, groupNames, groupColors, threshold,
  geneNames, geneTs, geneMs)
```

Arguments

`x`	An object of class `fsa` to which add a model.
`model`	A `list` aggregating the model parameters. Can be provided instead of the argument vectors below individually.
`groupMeans`	Numeric vector of length 2, the means of the scores in each group as computed on a training series. If `model` is provided, this vector will be used instead.
`groupSDs`	Numeric vector of length 2, the standard deviations of the scores in each group as computed on a training series. If `model` is provided, this vector will be used instead.

groupNames	Character vector of length 2, the names of the group described in groupMeans, groupSDs and groupColors. If model is provided, this vector will be used instead.
groupColors	Character vector of length 2, the colors to use to plot each group (see par for allowed values). If model is provided, this vector will be used instead.
threshold	Single numeric vector, the confidence threshold to use for prediction (a call will be made only if it is at least at this level of certainty). If model is provided, this vector will be used instead.
geneNames	Character vector, the names of the genes whose expression is to be used. If model is provided, this vector will be used instead.
geneTs	Numeric vector, for each gene in geneNames, the statistic of a t.test comparing its expression between the two groups in a training series. If model is provided, this vector will be used instead.
geneMs	Numeric vector, for each gene in geneNames, the mean expression in the whole training series. If model is provided, this vector will be used instead.

Value

Returns an S3 object of class fsaModel.

Author(s)

Sylvain Mareschal

See Also

classify

Examples

```
# Example FSA file provided
fsa <- read.fsa(system.file("extdata/fsa_GEP/A5918.fsa", package="FSAtools"))

# Add model from design file
design <- designFile(system.file("extdata/design_GEP.conf", package="FSAtools"))
fsa <- add.model(fsa, model=design$GLOBALS$MODEL)

# Observe model
print(attr(fsa, "model"))
plot(attr(fsa, "model"))
```

`align.fsa` *Aligns peaks using size ladder*

Description

This function adds to a `fsa` object a linear regression model allowing the raw time indexes to be converted into base pair sizes, using a known size markers ladder.

Usage

```
 align.fsa(x, channel = "ROX", fullLadder = c(50, 60, 90, 100, 120, 150, 160, 180, 190,
  200, 220, 240, 260, 280, 290, 300, 320, 340, 360, 380, 400), useLadder = c(50, 60,
   90, 100, 120), outThreshold = 0.15, noiseLevel = 10, surePeaks = 5,
leakingRatios = c(-1, 10), trim = c("forward", "backward", "none"),
maskOffScale = FALSE, rMin = 0.999, rescue = FALSE, ylim = NULL, ...)
```

Arguments

`x`	An object of class `fsa`, as returned by `read.fsa`
`channel`	Single character value, the name of the channel used for size markers.
`fullLadder`	Integer vector, the size markers used in the assay (in base pairs).
`useLadder`	Integer vector, the size markers to use for the alignment (using only size markers nearing the expected size for the experimental peaks may achieve a more precise alignment). They must be present in `fullLadder`. If `NULL`, `fullLadder` will be used entirely.
`outThreshold`	Single numeric value, maximal distance from the computed size-marker intensity for a peak to be considered as a size-marker. If lower than 1, it is considered as a proportion of the size-marker intensity computed from sure peaks.
`noiseLevel`	Single numeric value, minimal intensity for a local maximum to be considered as a peak.
`maskOffScale`	Single logical value, whether to mask indexes with off-scale values in any channel to limit side-effects or not.
`surePeaks`	Single integer value, amount of peaks to use to compute size-marker intensity. They are selected at the end of the profile, as most artefacts are observed ate the beginning. Consider to reduce this value if your assay was prematurely ended.
`leakingRatios`	Numeric vector of length two, defining the thresholds to consider a marker peak as a leakage from another channel. A leakage is expected to show at least one channel with a negative value below the absolute value of the marker channel (first ratio, -1) and another channel with a positive value higher than 10 times the absolute value of the marker channel (second ratio, 10).
`trim`	Single character value, defining how to behave when more/less peaks than expected are read. "forward" will keep first peaks and adjust discarding the last ones, "backward" will keep last peaks and adjust discarding the first ones, and "none" will generate an error.

rMin	Single numeric value, minimum adjusted r squared value (see summary.lm) to consider an alignment as "good". Poor alignments raise a warning, and may be due to artefactual peaks in the size-marker channel or errors in fullLadder definition. Consider lowering outThreshold and raising noiseLevel to minimize artefact selection.
rescue	Single logical value, whether to plot a "rescue" profile or not. Rescue profiles are calls to plot.fsa on which diverse additionnal data is drawn to help diagnose alignment problems.
ylim	To be passed to plot.fsa for the alignment rescue plot, if enabled (see rescue).
...	Further arguments to be passed to plot.fsa for the alignment rescue plot, if enabled (see rescue).

Value

Returns the object of class fsa provided with updated attributes :

ladderModel	A numeric vector of linear regression coefficients to use to convert raw indexes into base pairs.
ladderExact	A named numeric vector of raw indexes at which size markers were detected.

Author(s)

Sylvain Mareschal, Philippe Ruminy

Examples

```
# Example FSA file provided
fsa <- read.fsa(system.file("extdata/fsa_GEP/A5918.fsa", package="FSAtools"))

# Plot subset of the profile (time index)
plot(fsa, units="index", xlim=c(4000,5000))

# Align using full ladder
fullLadder <- c(
  50, 60, 90, 100, 120, 150, 160, 180, 190, 200, 220,
  240, 260, 280, 290, 300, 320, 340, 360, 380, 400
)
fsa <- align.fsa(fsa, fullLadder=fullLadder)

# Plot subset of the profile (base pairs)
plot(fsa, units="bp", xlim=c(80,130))
```

`classify` *Apply the binary predictor to FSA peaks*

Description

Predict to which class the sample is most likely to belong, using a modified LPS model.

Usage

```
classify(x, plot = TRUE)
```

Arguments

`x`	A `fsa` object with `peaks` and `model` attributes.
`plot`	Single logical value, whether to plot a visual representation of the prediction or not.

Value

Returns a list :

`score`	The raw score used to make the prediction.
`p`	The probability to belong to each of the two groups.
`class`	The final prediction, as a group name. May be `NA` if no probability passes the model threshold.

Author(s)

Sylvain Mareschal

References

Radmacher MD, McShane LM, Simon R. *A paradigm for class prediction using gene expression profiles.* J Comput Biol. 2002;9(3):505-11.

Wright G, Tan B, Rosenwald A, Hurt EH, Wiestner A, Staudt LM. *A gene expression-based method to diagnose clinically distinct subgroups of diffuse large B cell lymphoma.* Proc Natl Acad Sci U S A. 2003 Aug 19;100(17):9991-6.

Bohers E, Mareschal S, Bouzelfen A, Marchand V, Ruminy P, Maingonnat C, Menard AL, Etancelin P, Bertrand P, Dubois S, Alcantara M, Bastard C, Tilly H, Jardin F. *Targetable activating mutations are very frequent in GCB and ABC diffuse large B-cell lymphoma.* Genes Chromosomes Cancer. 2014 Feb;53(2):144-53.

See Also

`read.fsa`, `peaks.fsa`, `add.model`, `generic.process`

Examples

```
# Example FSA file provided
fsa <- read.fsa(system.file("extdata/fsa_GEP/A5918.fsa", package="FSAtools"))
fsa <- align.fsa(fsa)

# Add model from design file
design <- designFile(system.file("extdata/design_GEP.conf", package="FSAtools"))
fsa <- add.model(fsa, model=design$GLOBALS$MODEL)

# Add peak heights
fsa <- peaks.fsa(fsa, peaks=design$GLOBALS$PEAKS)

# Classify sample
fsa <- classify(fsa, plot=TRUE)
print(attr(fsa, "classification"))
```

`designFile` *Process interface's design file*

Description

This function is a slave for `generic.process`. It process a design file and returns its processed elements as a list.

Usage

```
designFile(fileName)
```

Arguments

`fileName`	Single character value, the path and name of a design file to process.

Details

Design files are text file split in multiple sections. Each section starts with a "[NAME]" line and ends when the next section begins. Lines starting with a # sign are ignored, as well as blank lines.

Standard sections refer to an existing R function (the section name is expected to match the function name in a case-sensitive manner), each line in the section setting an argument to call this function : the first value is the argument name, then after a tabulation come one or many values separated by tabulations. Multiple values will be aggregated into a vector, and `type.convert` will try to guess the correct type.

One or many modifiers can be added in the section name after the function name followed by the : sign and separated by commas (e.g. `file.remove:first,nowarn`). Modifiers have the following effects :

first The function will be called only once, while processing the first .fsa file.

last The function will be called only once, while processing the last .fsa file.

nowarn The function will be embedded inside `suppressWarnings` to silently ignore warnings.

`$NAME` and `@NAME` can be used while setting an argument to refer to global variable `NAME`. Use `$` reference for character variables, which will be replaced with `gsub` and can thus be combined (e.g. `$NAME.txt`). Use `@` reference to obtain the raw R variable, regardless of its type. Default globals are :

FILE_PATH The full name and path of the .fsa file currently considered in the loop.

FILE_DIR The parent directory of the .fsa file currently considered in the loop.

FILE_NAME The base name (without path) of the .fsa file currently considered in the loop.

OBJECT The last object of class 'fsa' returned by any function called in the pipeline.

OUTPUT_PATH The full name and path defined by `output` while calling `generic.process`.

OUTPUT_DIR The parent directory of `output` while calling `generic.process`.

OUTPUT_NAME The base name (without path) defined by `output` while calling `generic.process`.

Sections with full uppercase names (only letters and _ are allowed) will define a new global variable with matching name. The global will be a named list, each line in the section defining a vector (the first value being the name for the vector in the list). The `table` modifier can be used while defining globals (e.g. `PEAKS:table`), to request instead the section to be parsed as a TSV file (one row in the section is one row in the table, columns are separated by tabulations, see `read.table`). The first row will be used as column names and the first column as row names.

A DESIGN global (`[DESIGN]` section) is strongly recommended to keep design file self-explained, the following elements are suggested :

author The name of the design author (for human readers only).

purpose The description of the design (for human readers only).

FSAtools Version of the FSAtools package for which the design was created (separated with dots).

updated Date of the last design update (YYYY-MM-DD).

Please refer to the two provided working examples to help building your own designs. Full processing with the examples are described in `generic.process` `examples` section, direct access to the example design files is shown below.

Value

Returns a multi-level `list`, with a direct children per function to call and an extra `GLOBALS` element.

Children are named according to the function to call, thus multiple children can have the same name.

Author(s)

Sylvain Mareschal

See Also

`generic.process`, `generic.interface`

Examples

```
# Example file provided
file <- system.file("extdata/design_GEP.conf", package="FSAtools")
design <- designFile(file)

# Alignment rescue design provided
file <- system.file("extdata/design_SNP.conf", package="FSAtools")
design <- designFile(file)
```

`export.attr` *Print an attribute of a 'fsa' object to a file*

Description

Adds the content of an attribute of a 'fsa' object to a CSV file, either appending new rows or columns.

Typically used in `generic.process` via the design file to export numeric data during the processing.

Usage

```
export.attr(x, attr, file, meta = character(0), sep = "\t", dec = ".", quote = TRUE)
```

Arguments

`x`	The `fsa` object whose attribute is to be printed.
`attr`	Single character value, the name of the attribute to print.
`file`	Single character value, the path and name to the file to create or update.
`meta`	Character vector, the names of `x` meta-data fields to export as extra columns.
`sep`	To be passed to `write.table`.
`dec`	To be passed to `write.table`.
`quote`	To be passed to `write.table`.

Details

New data will be appended below the content of the file with extra 'meta' columns, row names will be added only if the file was empty.

Value

Invisibly returns `TRUE` on success.

Author(s)

Sylvain Mareschal

See Also

`read.fsa`, `peaks.fsa`, `genotype.closest.fsa`, `genotype.ratio.fsa`, `classify`

filter.fsa *Applies filter() to a "fsa" object*

Description

Replaces the requested column of values by the output of filter, possibly after masking values out of a specified index or bp range.

Usage

```
filter.fsa(x, channel, ..., from = NA, to = NA, units = "bp")
```

Arguments

x	An object of class fsa, as returned by read.fsa
channel	Single character value, the name of the channel used for size markers.
...	Further arguments to be passed to filter.
from	Single numeric value, the starting offset (integer index or numeric bp) to consider. No subsetting will be applied if NA.
to	Single numeric value, the last offset (integer index or numeric bp) to consider. No subsetting will be applied if NA.
units	Either "index" or "bp", defining the unit of from and to. Notice x must have been processed by align.fsa to use "bp".

Value

Returns x, with updated content.

Author(s)

Sylvain Mareschal

See Also

read.fsa

Examples

```
# Example FSA file provided
fsa <- read.fsa(system.file("extdata/fsa_GEP/A5918.fsa", package="FSAtools"))
fsa <- align.fsa(fsa)

# Profile before filtering
plot(fsa)

# Plot subset of the profile (base pairs)
fsa <- filter.fsa(fsa, channel="ROX", filter=20, from=40, to=140, units="bp")
```

```
# Profile after filtering
plot(fsa)
```

`fusions.process` *LD-RTPCR fusion identification by Sanger*

Description

Automatically interpret gene fusions found by Sanger sequencing using the Ligation-Dependent PCR protocol.

Usage

```
fusions.process(input, design, output, sheet = NA, cores = NA, ...)
```

Arguments

`input`	Single character value, the path to the directory containing the AB1 files to process.
`design`	Data.frame describing all possible fusions (see Details).
`output`	Single character value, the path to a directory in which to produce output files (will be created if doesn't yet exists).
`sheet`	Single character value, the name and path of a CSV file describing the files to process. 3 columns are expected: `ID` which gives a simpler sample name to use in outputs, `way` which defines if sequencing was 'forward' or 'reverse', and `file` which gives the file name and path relative to the `input` argument.
`cores`	Single integer value, the amount of CPUs to use on the local machine to parallelize the computation. If `NA`, a guess will be made. If 1, computation will not use the `parallel` package at all but only loop over samples.
`...`	Further arguments are passed to `fusions.process.core`.

Details

`design` must contain one row for each possible combination of a left primer with a right primer, whether this fusion is expected and relevant or not.

Expected columns in `design` are (excluding extra columns required with `extra`) :

left.name Character, the name of the left primer.

left.seq Character (uppercase), the sequence of the left primer (gene-specific part only).

left.unileft Character (uppercase), the sequence of the left universal primer used for amplification.

left.symbol Character, the symbol of the gene targeted by the left primer.

left.GRCh38 Character, the genomic coordinates of the last base of the left primer (chromosome:position:strand).

left.GRCh38_band Character, the cytogenetic location of the gene targeted by the left primer.

right.name Character, the name of the right primer.

right.seq Character (uppercase), the sequence of the right primer (gene-specific part only).

right.uniright Character (uppercase), the sequence of the right universal primer used for amplification.

right.symbol Character, the symbol of the gene targeted by the right primer.

right.GRCh38 Character, the genomic coordinates of the last base of the right primer (chromosome:position:strand).

right.GRCh38_band Character, the cytogenetic location of the gene targeted by the right primer.

seq_forward Character (uppercase), the complete sequence expected in forward sequencing (concatenation of `left.unileft`, `left.seq`, `right.seq`, `right.uniright` and the right tail, if any).

seq_reverse Character (uppercase), the complete sequence expected in reverse sequencing (reverse complement of a concatenation of the left tail, if any, `left.unileft`, `left.seq`, `right.seq`, `right.uniright`).

Please contact the authors to obtain a relevant design object.

Value

Invisibly returns the aggregated table of top results for all samples.

Various files are produced, in location set by the `output` argument :

Top.csv The aggregated table of top results for all samples.

***.pdf** One plot for each sample, showing the sequencing profile and the best alignments found.

Author(s)

Sylvain Mareschal

See Also

`generic.process`

`generic.log` *Wrapper for* `generic.process`

Description

This function is mainly a wrapper for `generic.process`, diverting messages, warnings and errors to a more readable log file.

Usage

```
generic.log(..., logFile)
```

Arguments

...	Arguments to be passed to generic.process.
logFile	Single character value, the path and name of the log file where to divert output.

Value

Either an error object if one occured, an integer number of warnings which happened during the (otherwise successfull) processing or TRUE if everything went fine.

Author(s)

Sylvain Mareschal

See Also

generic.process, generic.interface

Examples

```
# Working in temporary directory
output <- sprintf("%s/GEP", tempdir())
logFile <- sprintf("%s.log", output)

# Direct analysis
generic.process(
  input = system.file("extdata/fsa_GEP", package="FSAtools"),
  design = system.file("extdata/design_GEP.conf", package="FSAtools"),
  output = output
)

# Logged analysis (check logFile)
generic.log(
  input = system.file("extdata/fsa_GEP", package="FSAtools"),
  design = system.file("extdata/design_GEP.conf", package="FSAtools"),
  output = output,
  logFile = logFile
)
```

---

generic.process *Processing multiple FSA files*

Description

generic.process handles the whole analysis of a series of .fsa files according to the pipeline described in the user-provided design file, generating tabular and graphical profiles.

generic.interface summons a Tcl-Tk interface to call generic.process interactively.

Usage

```
generic.process(input, design, output, include = NULL, exclude = NULL,
   progressBar = NULL)
generic.interface()
```

Arguments

input	Single character value, the path to a directory containing .fsa files to analyse. Notice it will be explored recursively, so sub-directories are allowed.
design	Single character value, the path to a design file, as handled by designFile.
output	Single character value, the path to a ".pdf" or ".log" file that will be created during the analysis.
include	Single character value, a regular expression files (with relative path) in input must match to be processed (ignored if NULL).
exclude	Single character value, a regular expression files (with relative path) in input must not match to be processed (ignored if NULL).
progressBar	A ttkprogressbar to increment during the processing, or NULL. This argument is only provided to connect GEP.interface and GEP.process, thus it should be ignored.

Details

The content of the analysis pipeline is fully controlled by the design file, see designFile for details and the examples section below for two working examples provided in the package.

More generally, generic.process loops over the list of .fsa files in the input directory and calls the requested functions one after the other, updating the fsa object at each step.

Value

Return nothing. generic.process raise errors, warnings and messages which are intercepted by generic.interface and redirected to the log file (output.log).

Author(s)

Sylvain Mareschal

References

Mareschal, Ruminy et al (2015) <doi:10.1016/j.jmoldx.2015.01.007> "Accurate Classification of Germinal Center B-Cell-Like/Activated B-Cell-Like Diffuse Large B-Cell Lymphoma Using a Simple and Rapid Reverse Transcriptase-Multiplex Ligation-Dependent Probe Amplification Assay: A CALYM Study"

See Also

designFile

Examples

```
### EXAMPLE 1 : Gene expression (RT-MLPA) ###

# Working in temporary directory
output <- sprintf("%s/GEP", tempdir())

# See files before analysis
dir(system.file("extdata", package="FSAtools"))

# Launch analysis in package directory
generic.process(
  input = system.file("extdata/fsa_GEP", package="FSAtools"),
  design = system.file("extdata/design_GEP.conf", package="FSAtools"),
  output = output
)

# List resulting files
dir(dirname(output), full.names=TRUE)


### EXAMPLE 2 : Genotyping ###

# Working in temporary directory
output <- sprintf("%s/SNP", tempdir())

# See files before analysis
dir(system.file("extdata", package="FSAtools"))

# Launch analysis in package directory
generic.process(
  input = system.file("extdata/fsa_SNP", package="FSAtools"),
  design = system.file("extdata/design_SNP.conf", package="FSAtools"),
  output = output
)

# List resulting files
dir(dirname(output), full.names=TRUE)
```

`genotype.fsa` *Calls alleles for a SNP genotyping experiment*

Description

Calls alleles in experiments where fragments of different sizes are expected according to the allele.

`genotype.closest.fsa` selects for each allele the closest expected normalized peak height among the 0, 1 and 2 copy values provided in the design.

`genotype.ratio.fsa` calls an allele as absent, heterozygous or homozygous using fixed thresholds on the proportion of signal from a locus.

genotype.N1.fsa calls an allele as present if it exceeds a given proportion of the expected signal for 1 copy, as provided in the design. When a single allele is called present for a locus, it is considered homozygous.

Usage

```
genotype.closest.fsa(x)
genotype.ratio.fsa(x, homo = 0.85, hetero = c(0.3, 0.7))
genotype.N1.fsa(x, threshold = 0.1)
```

Arguments

x	The fsa object to use, which must have a peaks attribute (see peaks.fsa). Peaks are expected to be named according to the LOCUS_ALLELE pattern. genotype.closest.fsa() expected optional N0, N1 and N2 columns with expected normalized heights for 0, 1 and 2 copies of the allele.
homo	Single numeric value, the ratio of the considered allele signal over all signal from this locus to call a homozygous allele. Similarly alleles below 1 - homo will be called "absent".
hetero	Numeric vector of length two, minimum and maximum ratios of the considered allele signal over all signal from this locus to call a heterozygous allele.
threshold	Single numeric value, the ratio of the considered allele signal over the expected value for 1n to call an allele.

Value

Returns x, with a new or updated genotypes attribute, a data.frame with a row for each locus :

call	The concatenation of the two called alleles, possibly with ?.
alleles	The comma-separated list of ratios observed for all alleles (genotype.ratio.fsa() only).

A calls vector attribute is also set, corresponding to the call column only.

Note

All three functions assume peaks to be named according to the following convention : "ALLELE - LOCUS".

Author(s)

Sylvain Mareschal

See Also

generic.process

Examples

```
# Using a design file
design <- designFile(system.file("extdata/design_SNP.conf", package="FSAtools"))

# Example FSA file provided
fsa <- read.fsa(system.file("extdata/fsa_SNP/A7840.fsa", package="FSAtools"))
fsa <- align.fsa(
  fsa,
  channel = design$align.fsa$channel,
  outThreshold = design$align.fsa$outThreshold,
  useLadder = design$align.fsa$useLadder
)
fsa <- peaks.fsa(fsa, peaks=design$GLOBALS$PEAKS)

# Genotype
fsa <- genotype.ratio.fsa(fsa)
print(attr(fsa, "genotypes"))
print(attr(fsa, "calls"))
```

`multiplot` *Wrapper to layout*

Description

Calls `layout` using atomic arguments and convenient defaults, mainly to be included in `generic.process` design files.

Usage

```
multiplot(nrow, ncol, widths = rep.int(1, ncol), heights = rep.int(1, nrow),
  indexes = 1:(nrow * ncol), byrow = FALSE, respect = FALSE, cex = 1)
```

Arguments

`nrow`	Single integer value, the amount of rows into which divide the screen for multiple plots.
`ncol`	Single integer value, the amount of columns into which divide the screen for multiple plots.
`widths`	To be passed to `layout`.
`heights`	To be passed to `layout`.
`indexes`	Integer vector, the ordering of plots while building the `mat` matrix for `layout`.
`byrow`	Single logical value, whether to fill the `mat` matrix for `layout` with `indexes` by row or by column.
`respect`	To be passed to `layout`.
`cex`	Single numeric value, if not `NA` `par` will be called to force this value, as large layout automatically change it.

Value

Invisibly returns TRUE on success.

Author(s)

Sylvain Mareschal

See Also

generic.process

Examples

```
multiplot(nrow=1, ncol=3, widths=c(1,2,2))
plot(1:5)
plot(1:5)
plot(1:5)
```

---

peaks.fsa *Get maximal value in ranges*

Description

Look for the maximal value in one or many ranges, typically for peak detection.

Usage

```
peaks.fsa(x, peaks, names, size.min, size.max, channels, colors,
  logTransform = FALSE, lowThreshold = 1000, noiseRange = c(-10, 0))
```

Arguments

x	An aligned object of class fsa, as returned by align.fsa.
peaks	A data.frame with one row for each peak to consider. Can be provided instead of individual names (corresponding to peaks row names instead of a column), size.min, size.max, channels and colors argument vectors.
names	Character vector, the names to give to the peaks (duplicated values should be avoided). If peaks is provided, this vector will be used instead.
size.min	Numeric vector, the minimal size (in base pairs) to look for the corresponding peak. If peaks is provided, this vector will be used instead.
size.max	Numeric vector, the maximal size (in base pairs) to look for the corresponding peak. If peaks is provided, this vector will be used instead.
channels	Character vector, the name of the channel in x in which to look for the corresponding peak. If peaks is provided, this vector will be used instead.
colors	Vector defining the color to use in future plots to highlight the corresponding peak. If peaks is provided, this vector will be used instead.

logTransform	Single logical value, whether to apply log transformation (base 2) to normalized values (previously floored to 0 and summed with 1) or not.
lowThreshold	Single numeric value, threshold for which "low profile" warnings are called if all peaks are lower.
noiseRange	Numeric vector of length 2, defining the range (relative to the starting range of the first peak defined in ranges) in which measure the noise (in bp). If the noise peak is 20 percent greater than the first peak, a warning is raised as the accuracy of the measure may be compromised.

Value

Returns x, with a new or updated peaks attribute, a data.frame with a row for each range :

size.min	User-provided argument.
size.max	User-provided argument.
channels	User-provided argument.
colors	User-provided argument.
size	Size at which the maximum was found, in base pairs.
height	Maximum found, in fluorescence units.
offScale	Is there any off-scale value in the range ?
normalized	Current peak's height divided by the mean of all peak heights.

A normalized vector attribute is also set, corresponding to the normalized column only.

Author(s)

Sylvain Mareschal

See Also

generic.process

Examples

```
# Example FSA file provided
fsa <- read.fsa(system.file("extdata/fsa_GEP/A5918.fsa", package="FSAtools"))
fsa <- align.fsa(fsa)

# Single custom interval
fsa <- peaks.fsa(
  fsa,
  names = "IRF4",
  size.min = 86.2,
  size.max = 87.5,
  channels = "6-FAM",
  colors = "blue"
)
print(attr(fsa, "peaks"))
```

```
# Using a design file
design <- designFile(system.file("extdata/design_GEP.conf", package="FSAtools"))
fsa <- peaks.fsa(fsa, peaks=design$GLOBALS$PEAKS)
print(attr(fsa, "peaks"))
```

`plot.fsa` *Plot method for "fsa" objects*

Description

Plots a `fsa` object. For each selected channel, a line is drawn bewteen measured fluorescence intensities (y axis) along the electrophoresis time (x axis).

Usage

```
## S3 method for class 'fsa'
plot(x, units = NA, channels = NA, chanColors = NA, ladder = TRUE,
    offScaleCol = "#FF0000", offScalePch = "+", offScaleCex = 0.4, bg = "white",
    fg = "black", title = "", title.adj = 0, title.line = NA, xlab = NA,
    ylab = "Intensity", xlim = NA, ylim = NA, xaxt = "s", yaxt = "s", bty = "o",
    xaxp = NA, nticks = 5, all.bp = TRUE, peaks.alpha = 48L, peaks.srt = 30,
   peaks.adj = c(0, 0), peaks.cex = 1.3, peaks.font = 2, legend.x = "topleft", ...)
```

Arguments

`x`	The `fsa` object to plot.
`units`	Single character value, the unit to use on x axis. "index" uses the raw index contained in files, "bp" usess base pair estimations but needs the object to be aligned first using `align.fsa`. `NA` will select "bp" if `x` is aligned, "index" elsewhere.
`channels`	Character or integer vector, the channels to plot. If `NA`, all channels are selected.
`chanColors`	Character vector defining colors to use to plot channels. Can be named according to channel names stored in `x`, or parallel with `channels` (first color for first channel, etc, no recycling). If `NA`, colors stored in `x` are used. See the `col` argument in `par` for further details on allowed values.
`ladder`	Single logical value, whether to add an x axis with size ladder peaks or not. Raises a warning if `x` was not aligned before plotting.
`offScaleCol`	To be passed to `points` for off-scale value plot (see `par` for allowed values).
`offScalePch`	To be passed to `points` for off-scale value plot (see `par` for allowed values).
`offScaleCex`	To be passed to `points` for off-scale value plot (see `par` for allowed values).
`bg`	See `par` for further details.
`fg`	See `par` for further details. This value is also used for col.axis, col.lab, col.main and col.sub graphical parameters.
`title`	Single character value, the main title to print on the plot.
`title.adj`	To be passed as `adj` to `title`.

title.line	To be passed as line to title.
xlab	See plot for further details. If NA, units is used.
ylab	See plot for further details.
xlim	See plot for further details. If NA, x range is used.
ylim	See plot for further details. If NA, x range is used.
xaxt	See par for further details.
yaxt	See par for further details.
bty	See par for further details.
xaxp	See par for further details. If NA, a suitable value is computed.
nticks	Single integer value. When xaxp is NA and units is "bp", this values fixes the interval between X axis labels.
all.bp	Single logical value, whether to force an unlabeled axis tick at each bp when units is "bp" or not.
peaks.alpha	Single integer value, the alpha channel to add to peak colors to make a background (255 is no transparency at all, 0 is invisible).
peaks.srt	To be passed as srt to text while printing peak names.
peaks.adj	To be passed as adj to text while printing peak names.
peaks.cex	To be passed as cex to text while printing peak names.
peaks.font	To be passed as font to text while printing peak names.
legend.x	To be passed as x to legend.
...	Further arguments to be passed to plot.

Value

Invisibly returns TRUE on success.

Author(s)

Sylvain Mareschal

See Also

read.fsa

Examples

```
# Example FSA file provided
fsa <- read.fsa(system.file("extdata/fsa_GEP/A5918.fsa", package="FSAtools"))

# Plot whole profile
plot(fsa)

# Plot subset of the profile (time index)
plot(fsa, units="index", xlim=c(4000,5000))
```

```
# Plot subset of the profile (base pairs)
fsa <- align.fsa(fsa)
plot(fsa, units="bp", xlim=c(80,130))
```

`plot.fsaModel` *Plot method for "fsaModel" objects*

Description

Plots a `fsaModel` object.

Usage

```
## S3 method for class 'fsaModel'
plot(x, xlab = "Score", lwd = 3, ...)
```

Arguments

`x`	The `fsaModel` object to plot.
`xlab`	To be passed to `plot`.
`lwd`	To be passed to `plot`.
`...`	Further arguments to be passed to `plot`.

Value

Invisibly returns `TRUE` on success.

Author(s)

Sylvain Mareschal

See Also

`train`

Examples

```
# Example FSA file provided
fsa <- read.fsa(system.file("extdata/fsa_GEP/A5918.fsa", package="FSAtools"))

# Add model from design file
design <- designFile(system.file("extdata/design_GEP.conf", package="FSAtools"))
fsa <- add.model(fsa, model=design$GLOBALS$MODEL)

# Plot model
plot(attr(fsa, "model"))
```

print.fsa *Print method for "fsa" objects*

Description

Prints a short summary of an fsa object.

Usage

```
## S3 method for class 'fsa'
print(x, ...)
```

Arguments

x	The fsa object to print.
...	Currently ignored.

Value

Invisibly returns TRUE on success.

Author(s)

Sylvain Mareschal

See Also

read.fsa

Examples

```
# Example FSA file provided
fsa <- read.fsa(system.file("extdata/fsa_GEP/A5918.fsa", package="FSAtools"))
print(fsa)

# Aligned version
fsa <- align.fsa(fsa)
print(fsa)
```

`read.abif` *Read ABIF formatted files*

Description

ABIF stands for Applied Biosystem Inc. Format, a binary format modeled after TIFF format. Corresponding files usually have an `*.ab1` or `*.fsa` extension.

Usage

```
read.abif(filename, max.bytes.in.file = file.info(filename)$size,
  pied.de.pilote = 1.2, verbose = FALSE)
```

Arguments

`filename`	The name of the file.
`max.bytes.in.file`	The size in bytes of the file, defaulting to what is returned by `file.info`
`pied.de.pilote`	Safety factor: the argument `n` to `readBin` is set as `pied.de.pilote*max.bytes.in.file`.
`verbose`	logical [FALSE]. If TRUE verbose mode is on.

Details

All data are imported into memory, there is no attempt to read items on the fly.

Value

A list with three components: `Header` which is a list that contains various low-level information, among which `numelements` is the number of elements in the directory and `dataoffset` the offset to find the location of the directory. `Directory` is a data.frame for the directory of the file with the number of row being the number of elements in the directory and the 7 columns describing various low-level information about the elements. `Data` is a list with the number of components equal to the number of elements in the directory.

Note

This function and the current help page were duplicated from the **seqinr** package in its 3.0-7 version (available on the CRAN under GPL 2 licensing).

Author(s)

J.R. Lobry, 'bool' type implemented by Sylvain Mareschal

References

Charif, D. and Lobry, J.R. (2007) *Structural approaches to sequence evolution: Molecules, networks, populations* ISBN 978-3-540-35305-8, pp 207-232.

Anonymous (2006) Applied Biosystem Genetic Analysis Data File Format. Available at https://projects.nfstc.org/workshops/resources/articles/ABIF_File_Format.pdf. Last visited on 2020-07-10.

See Also

`readBin` which is used here to import the binary file and `file.info` to get the size of the file.

Examples

```
# Example FSA file provided
rawFsa <- read.abif(system.file("extdata/fsa_GEP/A5918.fsa", package="FSAtools"))
```

`read.fsa` *Imports a .fsa file from Applied Biosystems*

Description

This function parses a FSA file holding fragment analysis data, using **seqinr** package's `read.abif`.

Usage

```
read.fsa(file, lowess = TRUE, lowess.top = 200, processed = FALSE, meta.extra = NULL,
   quiet = FALSE, ...)
```

Arguments

`file`	Single character value, the name and path of the file to parse.
`lowess`	Single logical value, whether to apply `lowess` on intensities to smooth time-related biases or not.
`lowess.top`	Single numeric value, values flagged as "off-scale" or above this threshold will be replaced by `lowess.top` to compute the lowess smooth, in order to limit the impact of high and wide peaks.
`processed`	Single logical value, whether to use processed DATA values (as stored in sets 9 to 12, not always available) rather than raw values (sets 1 to 4). If `NA`, processed ones will be used as long as they are available, else raw ones will be used instead.
`meta.extra`	Named character vector, defining which extra fields to extract to populate the `runMetaData` attribute. The vector names define the human-readable names to use in output, the vector values provide the 4 uppercase letter code to extract (all values will be gathered in a vector if the code is used several times). See the reference provided in `read.abif` for existing codes in the ABIF file format.
`quiet`	Single logical value, whether to print FSA meta-data read from the file or not.
`...`	Further arguments to be passed to `read.abif`.

Value

A S3 object of class `fsa`

Author(s)

Sylvain Mareschal

See Also

`read.abif`, `generic.process`, `plot.fsa`, `read.sanger`

Examples

```
# Example FSA file provided
fsa <- read.fsa(system.file("extdata/fsa_GEP/A5918.fsa", package="FSAtools"))
print(fsa)
```

`read.sanger` *Imports a .ab1 file from Applied Biosystems corresponding to Sanger sequencing*

Description

This function parses a FSA/AB1 file using `read.fsa`, with few adjustments for Sanger sequencing experiments.

Usage

```
read.sanger(file, channelOrder = NULL, guess.threshold = 0.3, processed = NA,
   lowess = FALSE, ...)
```

Arguments

`file`	Single character value, the name and path of the file to parse.
`channelOrder`	Character vector, providing 'A', 'C', 'G' and 'T' in the order of the used channels. If `NULL`, a guess will be attempted based on the called sequence.
`guess.threshold`	Single numeric value, setting the tolerance to use for channel guessing validation. Lower values mean higher chances to get an error for channel guessing failure.
`processed`	To be passed to `read.fsa`.
`lowess`	To be passed to `read.fsa`.
`...`	To be passed to `read.fsa`.

Value

A S3 object of class `fsa`

Author(s)

Sylvain Mareschal

See Also

read.fsa, read.abif

---

train *Training function for binary predictors*

Description

This function build a model from data to predict class in classify.

Usage

```
train(peakMatrix, group, filter.p = 0.05, groupColors = c("red", "blue"),
  threshold = 0.9)
```

Arguments

peakMatrix	Numeric matrix of normalized peak heights with samples in rows and peaks in columns.
group	Two-level factor defining the group of every samples in peaks.
filter.p	Single numeric value, if not NA only genes for which the t-test p is lower than this will be used in the model.
groupColors	Vector of length two, defining the colors to use to represent the two groups in future plots.
threshold	Single numeric value, the likelihood threshold above which make a call when classifying (classification will return NA is this threshold is met for none of the two groups).

Value

Returns an S3 object of class fsaModel.

Author(s)

Sylvain Mareschal

See Also

add.model, classify

Examples

```
# Underlying truth for pseudo-data (10 genes)
geneNames <- paste("gene", LETTERS[1:10], sep=".")
geneMean <- abs(rnorm(10))
groupShift <- rnorm(10, sd=0.1)

# Generate pseudo-data for 50 samples
mtx <- NULL
for(g in 1:10) {
  x <- rnorm(n=50, mean=geneMean[g], sd=0.1)
  x[1:25] <- x[1:25] + groupShift[g]
  x[26:50] <- x[26:50] - groupShift[g]
  mtx <- cbind(mtx, x)
}
colnames(mtx) <- geneNames
rownames(mtx) <- c(
  paste("group1", 1:25, sep="."),
  paste("group2", 26:50, sep=".")
)

# Train model
group <- c(
  rep("group1", 25),
  rep("group2", 25)
)
model <- train(mtx, group)
plot(model)

# Compare model to truth
i <- match(geneNames, model$geneNames)
out <- data.frame(
  gene = geneNames,
  true.M = geneMean,
  model.M = model$geneMs[i],
  true.shift = groupShift,
  model.T = model$geneTs[i]
)
print(out)
```

---

wav2RGB *Converts light wavelengths to RGB colors*

Description

Converts wavelengths in nanometers into corresponding visible colors.

Usage

```
wav2RGB(wav)
```

Arguments

wav	Numeric vector of wavelengths (in nanometers) to convert into colors.

Value

Returns a character vector of the same length as wav, with an RGB color for each wavelength. Wavelengths out of visible ranges return black.

Author(s)

Sylvain Mareschal

References

http://codingmess.blogspot.fr/2009/05/conversion-of-wavelength-in-nanometers.html

Examples

```
wv <- seq(from=300, to=800, by=10)
plot(x=wv, y=rep(1, length(wv)), col=wav2RGB(wv), pch=19)
```

Index