# Package 'MLPA'

May 1, 2020

**Type** Package

**Title** Multiplex Ligation-Dependent Probe Amplification Data Analysis

**Version** 1.10.0

**Date** 2020-04-27

**URL** http://bioinformatics.ovsa.fr/MLPA

**BugReports** https://github.com/maressyl/R.MLPA/issues

**Description**
Functions to import Applied Biosystems data files of multiplex ligation-dependent probe amplification (MLPA) analysis and process them. Gene-expression profiling methods are described in Mareschal, Ruminy et al (2015) <doi:10.1016/j.jmoldx.2015.01.007>. Gene-fusion detection methods are described in Mareschal, Palau et al (under review).

**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 <maressyl@gmail.com>

**Repository** CRAN

**Date/Publication** 2020-05-01 18:20:03 UTC

## R topics documented:

---

## `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,
   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. |
| `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/A5918.fsa", package="MLPA"))

# 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(peaks, model, plot = TRUE)
```

### Arguments

| | |
|---|---|
| `peaks` | A `data.frame`, as returned by `peaks.fsa`. |
| `model` | A `fsaModel` object, as returned by `model`. |
| `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`, `model`, `GEP.process`

## Examples

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

# Build model from design file
design <- designFile(system.file("extdata/design.conf", package="MLPA"))
design$model$disable <- NULL
model <- do.call("model", design$model)

# Get peak heights
peaks <- peaks.fsa(fsa, ranges=design$PEAKS$ranges)

# Classify sample
p <- classify(peaks, model, plot=TRUE)
print(p)

# Observe sample in model
plot(model)
abline(v=p$score)
```

---

## `designFile` *Process interface's design file*

---

### Description

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

### Usage

```
designFile(fileName, overwrite = list())
```

### Arguments

| | |
|---|---|
| `fileName` | Single character value, the path and name of a design file to process. |
| `overwrite` | Two-level named list as returned by `designFile`, values from this list will replace parameters parsed from the design file. |

### Details

Design files are tab-separated files, with a header line and dots as decimal separators. It can be quoted, with double or simple quotes. "#" are considered as comment markup.

Design files are plain text files, separated in multiple sections. Sections begin with a "[NAME]" line and end when the next section begin. Comments (line beggining with the "#" character) and blank lines can be added everywhere, except into the PEAKS table. Quotes should be avoided, and multiple values for a single parameter (vectors) can be obtained by concatenating multiple values separated by commas (no spacing).

The "[DESIGN]" section contain name/value pairs, separated by tabulations :

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

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

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

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

The "[PEAKS]" section contain a tab-separated table, with a header line and dots as decimal separators. It should contain one row for each peak that is to be measured in the assay.

**name** Character, the name of the gene described (mist be unique in the table).

**channel** Character, the name of the channel in which a measure has to be done.

**size.min** Numeric, the start of the range in which a measure has to be done. For size markers, the range is defined in time index units, for genes in base pairs (after an `align.fsa` call).

**size.max** Numeric, the end of the range in which a measure has to be done. For size markers, the range is defined in time index units, for genes in base pairs (after an `align.fsa` call).

**color** Character, the color tu use for the range on the plot. This can be an english color name, or an hexadecimal color specification as "#000000". Notice transparency will be added.

Function-specific sections may also be present, to define R function arguments. Currently `read.fsa`, `align.fsa`, `plot.fsa`, `model` and `classify` functions are handled. Section names are supposed to respect function name case, surrounded by square brackets. These sections should contain name/value pairs, separated by tabulations, using only valid function arguments as names (please refer to the corresponding help page). Arguments not defined in the design file will be set to the function's default (raising a warning), and arguments not used by the function will be ignored (raising a warning too). An additionnal `disable` argument is handled (single logical value), to disable the call to the corresponding function.

## Value

Returns a multi-level `list`, with a direct children per section.

"DESIGN" directly transcribes as a named and typed `list` the elements described above.

"PEAKS" contains a list with the following elements :

| | |
|---|---|
| `ranges` | The concatenated `size.min` and `size.max` columns, for genes. |
| `channels` | The `channel` column, for genes. |
| `weights` | The `weigh` column, for genes. |
| `colors` | The plain color from `color` column, for genes. |
| `backgrounds` | The transparent version of the `color` column, for genes. |

Function-specific sections directly transcribe arguments as a named and typed `list`.

## Note

It is highly recommended to new users to build their design modifying "extdata/design.conf", updating the "[model]" and "[PEAKS]" sections.

Replacing the content of the "[model]" and "[classify]" sections by "disable TRUE" is also a good way to start with designs, as the classification model is optionnal and can be added later once the technique is developped.

The "extdata/recue.conf" design can prove particularly useful when processing fails with a non-obvious message (generally because the alignment failed). Processing files with this design disables almost everything and provides a raw profile that can help disgnose the problem.

### Author(s)

Sylvain Mareschal

### See Also

`GEP.interface`

### Examples

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

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

---

## `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, sheet = NA, output = ".", 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). |
| `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. |
| `output` | Single character value, the path to a directory in which to produce output files (will be created if doesn't yet exists). |
| `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

`GEP.process`

`GEP.process` *MLPA peak detection*

## Description

`GEP.process` handles the whole analysis from .fsa files, generating tables of expression values and graphical profiles.

`GEP.interface` summons a Tcl-Tk interface to call `GEP.process` interactively.

## Usage

```
GEP.process(input, design, output, overwrite = list(), gene.cex = 1.3,
  file.line = 3, mar = c(5,4,5,1), progressBar = NULL)
GEP.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. |
| `overwrite` | Named list, to be passed to `designFile()`. |
| `gene.cex` | Single numeric value, the character expanding factor for gene names. |
| `file.line` | Single numeric value, the line on which print the file name on plots. |
| `mar` | Numeric vector with 4 values, the margin sizes on bottom, left, top and right sides respectively. |
| `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. |

## Value

Return nothing. `GEP.process` raise errors, warnings and messages which are intercepted by `GEP.interface` and redirected to the log file.

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

**~.expr.tsv** Numeric matrix of normalized expressions (each sample is divided by its means).

**~.peaks.tsv** Table collecting all the peaks called during the analysis, with their size, intensity, sample, annotation and off-scale status.

**~.pdf** Profiles of the samples analysed. See `designFile` for customisation.

**~.log** Log file of errors, warnings and message (only `GEP.interface` produces it).

### 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

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

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

# Launch analysis in package directory
GEP.process(
  input = system.file("extdata/fsa", package="MLPA"),
  design = system.file("extdata/design.conf", package="MLPA"),
  output = output
)

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

## `model` *Object constructor for binary predictors*

### Description

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

### Usage

```
model(groupMeans, groupSDs, groupNames, groupColors = c("blue", "red"),
  threshold = 0.9, geneNames, geneTs, geneMs)
```

## Arguments

| | |
|---|---|
| `groupMeans` | Numeric vector of length 2, the means of the scores in each group as computed on a training series. |
| `groupSDs` | Numeric vector of length 2, the standard deviations of the scores in each group as computed on a training series. |
| `groupNames` | Character vector of length 2, the names of the group described in `groupMeans`, `groupSDs` and `groupColors`. |
| `groupColors` | Character vector of length 2, the colors to use to plot each group (see `par` for allowed values). |
| `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). |
| `geneNames` | Character vector, the names of the genes whose expression is to be used. |
| `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. |
| `geneMs` | Numeric vector, for each gene in `geneNames`, the mean expression in the whole training series. |

## Value

Returns an S3 object of class `fsaModel`.

## Author(s)

Sylvain Mareschal

## See Also

`classify`

## Examples

```
# Build from design file
design <- designFile(system.file("extdata/design.conf", package="MLPA"))
design$model$disable <- NULL
model <- do.call("model", design$model)

# Observe model
print(model)
plot(model)
```

## 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, ranges, logTransform = FALSE, lowThreshold = 1000, channels = "6-FAM",
  noiseRange = c(-10, 0), primerRange = c(35, 45))
```

### Arguments

| | |
|---|---|
| `x` | An aligned object of class `fsa`, as returned by `align.fsa`. |
| `ranges` | A named list of ranges, numeric vectors of length two (minimal and maximal values). They defines the sizes (in base pairs) for which a maximum is required. |
| `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. |
| `channels` | Single character value, the name of the `x` channel to browse. |
| `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. |
| `primerRange` | Numeric vector of length 2, defining the range in which measure primer signals (in bp). This is implemented for QC experimentation and may not be useful in current practice. |

### Value

Returns a `data.frame` with a row for each range :

| | |
|---|---|
| `gene` | The name of the range described, extracted from `ranges`. |
| `size.min` | Minimal size of the range described. |
| `size.max` | Maximal size of the range described. |
| `peak.size` | Size at which the maximum was found, in base pairs. |
| `peak.height` | Maximum found, in fluorescence units. |
| `peak.offScale` | Is there any off-scale value in the range ? |
| `normalized` | Current peak's height divided by the mean of all peak heights. |

### Author(s)

Sylvain Mareschal

## See Also

GEP.process

## Examples

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

# Single interval
print(peaks.fsa(fsa, ranges=list(IRF4=c(86.2, 87.5))))

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

---

## 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 = "", xlab = NA, ylab = "Intensity", xlim = NA, ylim = NA,
    xaxt = "s", yaxt = "s", bty = "o", xaxp = NA, nticks = 5, all.bp = TRUE, ...)
```

### 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. |
| `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. |
| `...` | Further arguments to be passed to `plot`. |

## Author(s)

Sylvain Mareschal

## See Also

`read.fsa`

## Examples

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

# 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`. |

### Author(s)

Sylvain Mareschal

### See Also

`train`

### Examples

```
# Build model from design file
design <- designFile(system.file("extdata/design.conf", package="MLPA"))
design$model$disable <- NULL
model <- do.call("model", design$model)

# Plot model
plot(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. |

### Author(s)

Sylvain Mareschal

### See Also

`read.fsa`

### Examples

```
# Example FSA file provided
fsa <- read.fsa(system.file("extdata/fsa/A5918.fsa", package="MLPA"))
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 `http://www6.appliedbiosystems.com/support/software_community/ABIF_File_Format.pdf`. Last visited on 2018-03-27.

## 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/A5918.fsa", package="MLPA"))
```

## 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, applyLowess = TRUE, processed = FALSE, meta.extra = NULL, ...)
```

### Arguments

| | |
|---|---|
| `file` | Single character value, the name and path of the file to parse. |
| `applyLowess` | Single logical value, whether to apply `lowess` on intensities to smooth time-related biases or not. |
| `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. |
| `...` | Further arguments to be passed to `read.abif`. |

### Value

A S3 object of class `fsa`

### Author(s)

Sylvain Mareschal

### See Also

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

### Examples

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

### 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. |

### 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, ...)
```

## 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. |
| `...` | Further arguments to be passed to `model`. |

## Value

Returns an S3 object of class `fsaModel`.

## Author(s)

Sylvain Mareschal

## See Also

`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