Package ‘FTICRMS’

February 19, 2015

Type Package
Title Programs for Analyzing Fourier Transform-Ion Cyclotron Resonance Mass Spectrometry Data
Version 0.8
Date 2009-08-20
Author Don Barkauskas
Maintainer Don Barkauskas <barkda@wald.ucdavis.edu>
Depends Matrix,lattice,splines
Description This package was developed partially with funding from the NIH Training Program in Biomolecular Technology (2-T32-GM08799).
License GPL-2
Repository CRAN
Date/Publication 2012-10-29 08:57:04
NeedsCompilation no

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTICRMS-package</td>
<td>2</td>
</tr>
<tr>
<td>baseline</td>
<td>2</td>
</tr>
<tr>
<td>display.tests</td>
<td>5</td>
</tr>
<tr>
<td>extract.pars</td>
<td>6</td>
</tr>
<tr>
<td>locate.peaks</td>
<td>9</td>
</tr>
<tr>
<td>make.par.file</td>
<td>10</td>
</tr>
<tr>
<td>run.all</td>
<td>13</td>
</tr>
<tr>
<td>run.analysis</td>
<td>14</td>
</tr>
<tr>
<td>run.baselines</td>
<td>17</td>
</tr>
<tr>
<td>run.cluster.matrix</td>
<td>19</td>
</tr>
<tr>
<td>run.lrg.peaks</td>
<td>22</td>
</tr>
<tr>
<td>run.peaks</td>
<td>24</td>
</tr>
<tr>
<td>run.strong.peaks</td>
<td>26</td>
</tr>
</tbody>
</table>

Index 29
**FTICRMS-package**

*Fourier Transform-Ion Cyclotron Resonance Mass Spectrometry (FT-ICR MS) Analysis*

**Description**

Contains programs for identifying baseline curves and peaks and for statistical analysis of FT-ICR MS data.

**Details**

- **Package:** FTICRMS
- **Type:** Package
- **Version:** 0.8
- **Date:** 2009-08-20
- **License:** GPL-2

This package was developed partially with funding from the NIH Training Program in Biomolecular Technology (2-T32-GM08799).

**Author(s)**

Don Barkauskas

Maintainer: Don Barkauskas (<barkda@wald.ucdavis.edu>)

**baseline**

*Calculate Baselines for Spectroscopic Data*

**Description**

Computes an estimated baseline curve for a spectrum using the “BXR algorithm,” a method of Xi and Rocke generalized by Barkauskas and Rocke.

**Usage**

```r
baseline(spect, init.bd = 1e-11, sm.par = 2, max.iter = 20, tol = 5e-8,
          sm.div = NA, sm.norm.by = c("baseline", "overestimate", "constant"),
          neg.div = NA, neg.norm.by = c("baseline", "overestimate", "constant"),
          rel.conv.crit = TRUE, zero.rm = TRUE, halve.search = FALSE)
```
Arguments

- **spect**: vector containing the intensities of the spectrum
- **init.bd**: initial value for baseline; default is flat baseline at median height
- **sm.par**: smoothing parameter for baseline calculation
- **sm.ord**: order of derivative to penalize in baseline analysis
- **max.iter**: convergence criterion in baseline calculation
- **tol**: convergence criterion; see below
- **sm.div**: smoothness divisor in baseline calculation
- **sm.norm.by**: method for smoothness penalty in baseline analysis
- **neg.div**: negativity divisor in baseline calculation
- **neg.norm.by**: method for negativity penalty in baseline analysis
- **rel.conv.crit**: logical; whether convergence criterion should be relative to size of current baseline estimate
- **zero.rm**: logical; whether to replace zeros with average of surrounding values
- **halve.search**: logical; whether to use a halving-line search if step leads to smaller value of function

Details

If the spectrum is given by \( y_i \), then the algorithm works by maximizing the objective function

\[
F(\{b_i\}) = \sum_{i=1}^{n} b_i - \sum_{i=2}^{n-1} A_{1,i} (b_{i-1} - 2b_i + b_{i+1})^2 - \sum_{i=1}^{n} A_{2,i} [\max\{b_i - y_i, 0\}]^2
\]

using Newton’s method (with embedded halving line search if halve.search == TRUE) using starting value \( b[1] = \text{init.bd}[1] \) for all \( i \). The middle term controls the smoothness of the baseline and the last term applies a “negativity penalty” when the baseline is above the spectrum.

The smoothing factor \( \text{sm.par} \) corresponds to \( A_1^* \) in Barkauskas (2009) and controls how large the estimated \( n \)th derivative of the baseline is allowed to be (for \( \text{sm.ord} = n \)). From a practical standpoint, values of \( \text{sm.ord} \) larger than two do not seem to adequately smooth the baseline because the Hessian becomes computationally singular for any reasonable value of \( \text{sm.par} \).

The parameters \( \text{sm.div} \), \( \text{sm.norm.by} \), \( \text{neg.div} \), and \( \text{neg.norm.by} \) determine the methods used to normalize the smoothness and negativity terms. The general forms are \( A_{1,i} = n^4 A_1^* M_i/p \) and \( A_{2,i} = 1/M_i/p \). Here, \( n = \text{length(spect)} \); \( p \) is \( \text{sm.div} \) or \( \text{neg.div} \), as appropriate; and \( M_i \) is determined by \( \text{sm.norm.by} \) or \( \text{neg.norm.by} \), as appropriate. Values of “baseline” make \( M_i = b_i' \), where \( b_i' \) is the currently estimated value of the baseline; values of “overestimate” make \( M_i = b_i' - y_i \); and values of “constant” make \( M_i = \sigma \), where \( \sigma \) is an estimate of the noise standard deviation.

The values of \( \text{sm.norm.by} \) and \( \text{neg.norm.by} \) can be abbreviated and both have default value “baseline”. The default values of NA for \( \text{sm.div} \) and \( \text{neg.div} \) are translated by default to \( \text{sm.div} = 0.5223145 \) and \( \text{neg.div} = 0.4210109 \), which are the appropriate parameters for the FT-ICR mass spectrometry machine that generated the spectra which were used to develop this package. It is distinctly
possible that other machines will require different parameters, and almost certain that other spectroscopic technologies will require different parameters; see Barkauskas (2009a) for a description for how these parameters were obtained.

If \( \text{zero.nrm == TRUE} \) and \( y_a, \ldots, y_{a+k} = 0 \), then these values of the spectrum are set to be \( (y_{a-1} + y_{a+k+1})/2 \). (For typical MALDI FT-ICR spectra, a spectrum value of zero indicates an erased harmonic and should not be considered a real data point.)

### Value

A list containing the following items:

- **baseline**: The computed baseline
- **iter**: The number of iterations for convergence
- **changed**: Numeric vector of length \( \text{iter} \) containing the number of indicator variables that switched value on each iteration
- **hs**: Numeric vector of length \( \text{iter} \) containing the number of halving line-searches done on each iteration

### Note

The original algorithm was developed by Yuanxin Xi and David Rocke. The code in this package was first adapted from a Matlab program by Yuanxin Xi, then modified to account for the new methodology in Barkauskas (2009a).

- \( \text{halve.search = FALSE} \) is recommended unless both \( \text{sm.norm.by == "constant" and neg.norm.by == "constant"} \).

### Author(s)

Don Barkauskas (<barkda@wald.ucdavis.edu>)

### References


### See Also

- `run.baselines`
**display.tests**  
*Display Full Test Information for Peaks*

**Description**
Displays full test information (not just $p$-values) for peaks generated by `run.analysis`.

**Usage**
```r
display.tests(sig.rows = "all", summ = "anova", tests,  
 form = parameter.list$form,  
 use.model = parameter.list$use.model, ...)
```

**Arguments**
- `sig.rows` numeric or character vector used to select rows of `sigs`; default value returns all significant tests
- `summ` either a function or string representing a function which can be applied to the output of `use.model` or "none"
- `tests` numeric or character vector used to select rows of `clust.mat`; default value returns the rows in `clust.mat` corresponding to the rows in `sigs[sig.rows,]`
- `form` formula for use in `lm`; default is the one that was used to generate the significant peaks
- `use.model` function or string representing a function; what test to apply to data
- `...` arguments to be passed to `use.model`

**Details**
If `use.model` in `run.analysis` evaluates to anything other than `t.test`, then the only thing reported on each peak by `run.analysis` is the $p$-value. This program takes a specified subset of the significant peaks and returns a list consisting of the models generated by `use.model` (if `summ = "none"`) or `summ` applied to those models. Typical values for `summ` include `anova` and `summary`. Although the program is designed to be used on significant peaks, by defining `tests` directly in the function call, you can access any of the peaks in `clust.mat`. If `tests` is defined in the function call, its value overrides anything specified by `sig.rows`.

**Value**
A list with components equal to the models or summaries for the requested peaks.

**Note**
`clust.mat` and `sig.mat` must be defined in the workspace for this program to work—for example, in the results file output by `run.analysis`.
Author(s)

Don Barkauskas (<barkda@wald.ucdavis.edu>)

See Also

run.analysis, anova, lm, t.test

extract.pars  Extract Parameters from File

Description

Extracts the parameters in the file specified by par.file and returns them in list form.

Usage

extract.pars(par.file = "parameters.RData", root.dir = ".")

Arguments

par.file  string containing name of parameters file
root.dir  string containing directory of parameters file to be extracted from

Details

Used by run.analysis to record all the parameter choices in an analysis for future reference.

Value

A list with the following components:

add.norm  logical; whether to normalize additively or multiplicatively on the log scale
add.par  additive parameter for "shiftedlog" or "glog" options for trans.method
align.fcn  function (and inverse) to apply to masses before (and after) applying align.method
align.method  alignment algorithm for peaks
base.dir  directory for baseline files
bhbysubj  logical; whether to look for number of large peaks by subject (i.e., combining replicates) or by spectrum
calc.all.peaks  whether to calculate all possible peaks or only sufficiently large ones
cor.thresh  threshold correlation for declaring isotopes
covariates  data frame containing covariates used in analysis

cluster.constant  parameter used in running cluster.method

cluster.method  method for determining when two peaks from different spectra are the same
extract.pars

FDR
False Discovery Rate in Benjamini-Hochberg test

FTICRMS.version
Version of FTICRMS that created file

form
formula used in use.model

gengamma.quantiles
whether to use generalized gamma quantiles when calculating large peaks

halve.search
whether to use a halving-line search if step leads to smaller value of function

isotope.dist
maximum distance for declaring isotopes

lrg.dir
directory for significant peaks file

lrg.file
name of file for storing large peaks

lrg.only
whether to consider only peaks that have at least one “large” peak; i.e., identified
by run.lrg.peaks

masses
specific masses to test

max.iter
convergence criterion in baseline calculation

min.spect
minimum number of spectra necessary for peak to be used in run.analysis

neg.div
negativity divisor in baseline calculation

neg.norm.by
method for negativity penalty in baseline analysis

norm.peaks
which peaks to use in normalization

norm.post.repl
logical; whether to normalize after combining replicates

normalization
type of normalization to use on spectra before statistical analysis

num.pts
number of points needed for peak fitting

oneside.min
minimum number of points on each side of local maximum for peak fitting

overwrite
whether to replace existing files with new ones

par.file
string containing name of parameters file

peak.dir
directory for peak location files

peak.method
method for locating peaks

peak.thresh
threshold for declaring large peak

pre.align
shifts to apply before running run.strong.peaks

pvalfcn
function to calculate p-values

R2.thresh
$R^2$ value needed for peak fitting

raw.dir
directory for raw data files

rel.conv.crit
whether convergence criterion should be relative to size of current baseline esti-
mate

repl.method
how to deal with replicates

res.dir
directory for result file

res.file
name for results file

root.dir
directory for parameters file and raw data directory

sm.div
smoothness divisor in baseline calculation
extract.pars

sm.norm.by  method for smoothness penalty in baseline analysis
sm.ord      order of derivative to penalize in baseline analysis
sm.par      smoothing parameter for baseline calculation
subs        subset of spectra to use for analysis
subtract.base whether to subtract calculated baseline from spectrum
tol         convergence criterion in baseline calculation
trans.method data transformation method
use.model   what model to apply to data
zero.rm     whether to replace zeros in spectra with average of surrounding values

Note

do.call(make.par.file, extract.pars()) recreates the original parameter file

See make.par.file for a summary of which programs use each of the parameters in the list.

Author(s)

Don Barkauskas (<barkda@wald.ucdavis.edu>)

References


See Also

make.par.file, run.analysis
locate.peaks

`locate.peaks` *Locate Peaks in a FT-ICR MS Spectrum*

**Description**

Locates peaks in FT-ICR MS spectra assuming that the peaks are roughly parabolic on the log scale.

**Usage**

`locate.peaks(peak.base, num.pts = 5, R2.thresh = 0.98, onside.min = 1, peak.method = c("parabola", "locmaxes"), thresh = -Inf)`

**Arguments**

- `peak.base`: numeric matrix with two columns containing the masses and the transformed spectrum intensities
- `num.pts`: minimum number of points needed to have a peak
- `R2.thresh`: minimum $R^2$ needed to have a peak
- `oneside.min`: minimum number of points needed on each side of the local maximum
- `peak.method`: how to locate peaks
- `thresh`: only local maxes that are larger than this will be checked to see if they are peaks

**Details**

If `peak.method` == "parabola", the algorithm works by locating local maxima in the spectrum, then seeing if any `num.pts` consecutive points with at least `oneside.min` point(s) on each side of the local maximum have a coefficient of determination ($R^2$) of at least `R2.thresh` when fitted with a quadratic. If, in addition, the coefficient of the squared term is negative, then this is declared a peak and the vertex of the corresponding parabola is located. The coordinates of the vertex give the components `Center_hat` and `Max_hat` in the return value. The `Width_hat` component is the negative reciprocal of the coefficient of the squared term.

If `peak.method` == "locmax", then the algorithm merely returns the set of local maxima larger than `thresh`, and the `Width_hat` component of the return value is `NA`.

**Value**

A data frame with columns

- `Center_hat`: estimated mass of peak
- `Max_hat`: estimated intensity of peak
- `Width_hat`: estimated width of peak
Note

An extremely large value for \texttt{width\_hat} most likely indicates a bad fit.
\texttt{peak\_method} can be abbreviated. Using \texttt{peak\_method = "locmax"} will vastly speed up the run-time, but may affect the quality of the analysis.

As noted in both papers in the References, a typical FT-ICR MS spectrum has far more peaks than can be accounted for by actual compounds. Thus, defining a good value of \texttt{thresh} will vastly speed up the computation without materially affecting the analysis.

Author(s)

Don Barkauskas (<barkda@wald.ucdavis.edu>)

References


See Also

run.peaks

\begin{verbatim}
make.par.file Create Parameter File for FT-ICR MS Analysis
\end{verbatim}

Description

Creates a file of parameters that can be read by the functions in the \texttt{FTICRMS} package

Usage

\begin{verbatim}
make.par.file(covariates, form, par.file = "parameters.RData", root.dir = ".", ...)\end{verbatim}

Arguments

covariates data frame with rownames given by raw data files with extensions (e.g., ".txt") stripped
form object of class \texttt{"formula"} to be used for testing using \texttt{covariates}
par.file string containing name of file
root.dir string containing location for file
... parameters whose default values are to be overwritten (see below)
Details

Creates a file with name given by par.file in directory given by root.dir which contains values for all of the parameters used in the programs in the FTICRMS package. The possible parameters that can be included in par.file, their default values, their descriptions, and the program(s) in which they are used are as follows:

add.norm = TRUE
add.par = 0
align.fcn = NA
align.method = "spline"
base.dir = paste(root.dir, "/Baselines", sep="")
bhbnb = FALSE
calc.all.peaks = FALSE
corthresh = 0.8
FDR = 0.1
FTICRMS.version = "0.8"
gengamma.quantiles = TRUE
halve.search = FALSE
isotope.dist = 7
lrg.dir = paste(root.dir, "/Large_Peaks", sep="")
lrg.file = "lrg_peaks.RData"
lrg.only = TRUE
masses = NA
max.iter = 20
min.spect = 1
neg.div = NA
neg.norm.by = "baseline"
norm.peaks = "common"
norm.post.repl = FALSE
num.pts = 5
oneside.min = 1
overwrite = FALSE
par.file = "parameters.RData"
peak.dir = paste(root.dir, "/All_Peaks", sep="")
peak.method = "parabola"
peak.thresh = 3.798194
pre.align = FALSE
pval.fcn = "default"
R2.thresh = 0.98
raw.dir = paste(root.dir, "/Raw_Data", sep="")
rel.conv.crit = TRUE
repl.method = "max"
res.dir = paste(root.dir, "/Results", sep="")
res.file = "analyzed.RData"
root.dir = "."

logical; whether to normalize additively or multiplicatively on
additive parameter for "shiftedlog" or "glog" options for the
function (and inverse) to apply to masses before (and after) applying
alignment algorithm for peaks
directory for baseline files
logical; whether to look for number of large peaks by subject or
logical; whether to use generalized gamma quantiles when calculating
logical; whether to use a halving-line search if step leads to small
maximum distance for declaring isotopes
directory for large peaks file
name of file for storing large peaks
logical; whether to consider only peaks that have at least one "large"
specific masses to test
convergence criterion in baseline calculation
minimum number of spectra necessary for peak to be used in normalization
negativity divisor in baseline calculation
method for negativity penalty in baseline analysis
which peaks to use in normalization
logical; whether to normalize after combining replicates
number of consecutive points needed for peak fitting
minimum number of points on each side of local maximum for peaks
logical; whether to replace existing files with new ones
string containing name of parameters file
directory for peak location files
method for locating peaks
threshold for declaring large peak
shifts to apply before running run.strong.peaks
function to calculate p-values; default is overall p-value of testing
R^2 value needed for peak fitting
directory for raw data files
whether convergence criterion should be relative to size of current
how to deal with replicates
directory for results file
name for results file
directory for parameters file and raw data
make.par.file

sm.div = NA
sm.norm.by = "baseline"
sm.ord = 2
sm.par = 1e-11
subs
subtract.base = FALSE
tol = 5e-8
trans.method = "shiftedlog"
use.model = "lm"
zero.rm = TRUE

smoothness divisor in baseline calculation
method for smoothness penalty in baseline analysis
order of derivative to penalize in baseline analysis
smoothing parameter for baseline calculation
subset of spectra to use for analysis
logical; whether to subtract calculated baseline from spectrum
convergence criterion in baseline calculation
data transformation method
what model to apply to data
whether to replace zeros in spectra with average of surrounding

Value

No value returned; the file par.file is simply created in root.dir.

Note

do.call(make.par.file, extract.pars()) recreates the original parameter file.

See the individual function help pages for each function for more detailed descriptions of the above parameters.

align.method, cluster.method, neg.norm.by, normalization, peak.method, sm.norm.by, and
trans.method can be abbreviated.

Author(s)

Don Barkauskas (<barkda@wald.ucdavis.edu>)

References


See Also

extract.pars
A wrapper that calls all six functions needed for a full analysis.

run.all(par.file = "parameters.RData", root.dir = ".")

Arguments

par.file: string containing the name of the parameters file
root.dir: string containing location of raw data directory and parameters file

Details

Requires par.file to be in place before starting—for example by creating it with make.par.file.
Calls (in order) run.baselines, run.peaks, run.lrg.peaks, run.strong.peaks, run.cluster.matrix, and run.analysis.

Note

The analysis described in Barkauskas et al. (2008) can be (approximately) reproduced using the following parameter values instead of the defaults:

```
add.par = 10
calc.all.peaks = TRUE
gengamma.quantiles = FALSE
max.iter = 30
neg.norm.by = "constant"
peak.thresh = 4
pval.fcn = function(x){anova(x)[2,5]}
rel.conv.crit = FALSE
sm.norm.by = "constant"
subtract.base = TRUE
zero.rm = FALSE
```

(It is only an approximate reproduction because the stopping criterion for baseline calculation used in the article turned out to be a poor one and is no longer supported in the package. This shouldn’t make a very large difference, however.)

Author(s)

Don Barkauskas (<barkda@wald.ucdavis.edu>)
References


See Also

make.par.file, run.baselines, run.peaks, run.lrg.peaks, run.strong.peaks, run.cluster.matrix, run.analysis

---

run.analysis

Test for Significant Peaks in FT-ICR MS by Controlling FDR

Description

Takes the file generated by `run.cluster.matrix` and tests the peaks using Benjamini-Hochberg to control the False Discovery Rate.

Usage

```r
run.analysis(form, covariates, FDR = 0.1, norm.post.repl = FALSE, norm.peaks = c("common", "all", "none"), normalization, add.norm = TRUE, repl.method = "max", use.model = "lm", pval.fcn = "default", lrg.only = TRUE, masses = NA, isotope.dist = 7, root.dir = ".", lrg.dir, lrg.file = lrg_peaks.RData, res.dir, res.file = "analyzed.RData", overwrite = FALSE, use.par.file = FALSE, par.file = "parameters.RData", bhbysubj = TRUE, subs, ...)
```

Arguments

- `form` object of class “formula” to be used by `use.model` for testing using covariates
- `covariates` data frame containing covariates used in analysis
- `FDR` False Discovery Rate in Benjamini-Hochberg test
- `norm.post.repl` logical; whether to normalize after combining replicates
- `norm.peaks` which peaks to use in normalization
### run.analysis

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>normalization</td>
<td>type of normalization to use on spectra before statistical analysis; kept for compatibility (see below)</td>
</tr>
<tr>
<td>add.norm</td>
<td>logical; whether to normalize additively or multiplicatively on the log scale</td>
</tr>
<tr>
<td>repl.method</td>
<td>function or string representing the name of a function; how to deal with replicates</td>
</tr>
<tr>
<td>use.model</td>
<td>function or string representing the name of a function; what test to apply to data</td>
</tr>
<tr>
<td>pval.fcn</td>
<td>function to extract ( p )-values; default is overall ( p )-value of test</td>
</tr>
<tr>
<td>lrg.only</td>
<td>logical; whether to consider only peaks that have at least one “large” peak; i.e., identified by run.lrg.peaks</td>
</tr>
<tr>
<td>masses</td>
<td>specific masses to test</td>
</tr>
<tr>
<td>isotope.dist</td>
<td>maximum distance for declaring isotopes</td>
</tr>
<tr>
<td>root.dir</td>
<td>directory for parameters file and raw data</td>
</tr>
<tr>
<td>lrg.dir</td>
<td>directory for large peaks file; default is <code>paste(root.dir, &quot;/Large_Peaks&quot;, sep = &quot;&quot;)</code></td>
</tr>
<tr>
<td>lrg.file</td>
<td>name of file to store large peaks in</td>
</tr>
<tr>
<td>res.dir</td>
<td>directory for results file; default is <code>paste(root.dir, &quot;/Results&quot;, sep = &quot;&quot;)</code></td>
</tr>
<tr>
<td>res.file</td>
<td>name for results file</td>
</tr>
<tr>
<td>overwrite</td>
<td>logical; whether to replace existing files with new ones</td>
</tr>
<tr>
<td>use.par.file</td>
<td>logical; if TRUE, then parameters are read from par.file in directory root.dir</td>
</tr>
<tr>
<td>par.file</td>
<td>string containing name of parameters file</td>
</tr>
<tr>
<td>bhbysubj</td>
<td>logical; whether to look for number of large peaks by subject (i.e., combining replicates) or by spectrum</td>
</tr>
<tr>
<td>subs</td>
<td>subset of spectra to use for analysis; see below</td>
</tr>
<tr>
<td>...</td>
<td>additional parameters to be passed to use.model</td>
</tr>
</tbody>
</table>

### Details

Reads in information from file created by `run.cluster.matrix` and creates a file named `res.file` in directory `res.dir` which contains the following variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>amps</td>
<td>matrix of transformed amplitudes of alignment peaks</td>
</tr>
<tr>
<td>bysubj.var</td>
<td>a vector which tells which rows of covariates are identified as the same subject</td>
</tr>
<tr>
<td>centers</td>
<td>matrix of calculated masses of alignment peaks</td>
</tr>
<tr>
<td>clust.mat</td>
<td>matrix of transformed amplitudes of peaks used in statistical testing</td>
</tr>
<tr>
<td>min.FDR</td>
<td>FDR level required to get at least one significant test given the starting set of peaks</td>
</tr>
<tr>
<td>sigs</td>
<td>matrix containing all tests which are significant under at least one scenario</td>
</tr>
<tr>
<td>which.sig</td>
<td>matrix containing all tests peaked</td>
</tr>
<tr>
<td>parameter.list</td>
<td>if use.par.file = TRUE, a list generated by <code>extract.pars</code>; otherwise not defined</td>
</tr>
</tbody>
</table>

### Value

No value returned; the file is simply created.
Note

If use.par.file == TRUE and other parameters are entered into the function call, then the parameters entered in the function call overwrite those read in from the file. Note that this is opposite from the behavior for **FTICRMS** versions 0.7 and earlier.

**norm.peaks** determines the peaks used for normalization: "common" normalizes each spectrum using the average peak height of the alignment peaks from that spectrum in amps; "all" normalizes each spectrum using the average peak height of all peaks in that spectrum.

Normalization is obsolete but is included for compatibility with previous versions of the package. The valid normalization schemes translate to the new scheme as follows: "common" is norm.post.repl = FALSE and norm.peaks = "common"; "postbase" is norm.post.repl = FALSE and norm.peaks = "all"; "postrepl" is norm.post.repl = TRUE and norm.peaks = "all"; and "none" is norm.peaks = "none" (and norm.post.repl = FALSE, although this value is irrelevant).

Replicates for the same subject are assumed to be determined by the unique values of **covariates$subj**. (Future implementations will allow for other methods of defining this.) To analyze replicates as independent samples, use repl.method = "none". This will also speed up the run time if there are no replicates in the data set.

The argument **subs** can be logical or numeric or character; if it is defined, then **covariates** is modified to **covariates[subs,drop=F]**.

If **masses** is not **NULL**, then the listed masses plus anything that could be in the first isotope.dist - 1 isotope peaks of each mass are tested.

If something other than the p-value for the overall test statistic is needed, then the user-defined function for **pval.fcn** should have the form **pval.fcn = function(x){...}**, where **x** is a model object of the type returned by **use.model**; and should have a return value of the desired p-value.

If **use.model** evaluates to **t.test**, then the difference between the two groups for each peak is recorded in **which.sig$delta** and **sigs$delta**; otherwise, these columns consist entirely of **NA** entries.

Each rowname of **sigs** and **which.sig** represents the range of masses that were used to form that peak. The columns of those objects give the p-value of the peaks in each row, the number of samples that had large peaks for each row, and the significance of each test, coded as

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>peak not eligible for B-H</td>
</tr>
<tr>
<td>0</td>
<td>peak eligible for B-H but not declared significant</td>
</tr>
<tr>
<td>1</td>
<td>peak declared significant</td>
</tr>
</tbody>
</table>

The “S” labels refer to the number of large peaks that were necessary for a row to be eligible. For example, the column labeled S5 in **sigs** used as its starting set of p-values all rows which had **which.sig$num.lrg >= 5**. If **bhbysubj == TRUE**, then the entries of **num.lrg** are obtained by going subject-by-subject and for each mass counting the number of subjects who had at least one spectrum with a large peak at that mass; otherwise, **num.lrg** for each mass is simply the total number of spectra that had a large peak at that mass.
run.baselines

Author(s)

Don Barkauskas (<barkda@wald.ucdavis.edu>)

References


See Also

run.strong.peaks

---

run.baselines

Calculate and Store Baselines for Spectroscopic Data

Description

Takes the spectra from files in raw.dir, calculates the baselines from them, and writes the results in the directory base.dir.

Usage

```r
run.baselines(root.dir = ".", raw.dir, base.dir, overwrite = FALSE,
use.par.file = FALSE, par.file = "parameters.RData",
sm.par = 1e-11, sm.ord = 2, max.iter = 20, tol = 5e-8,
sm.div = NA, sm.norm.by = c("baseline", "overestimate", "constant"),
neg.div = NA, neg.norm.by = c("baseline", "overestimate", "constant"),
rel.conv.crit = TRUE, zero.rm = TRUE, halve.search = FALSE)
```

Arguments

- **root.dir**: directory for parameters file and raw data
- **raw.dir**: directory for raw data files; default is paste(root.dir, "/Raw_Data", sep = "")
- **base.dir**: directory for baseline files; default is paste(root.dir, "/Baselines", sep = "")
- **overwrite**: logical; whether to replace existing files with new ones
- **use.par.file**: logical; if TRUE, then parameters are read from par.file in directory root.dir
- **par.file**: string containing name of parameters file
- **sm.par**: smoothing parameter for baseline calculation
run.baselines

sm.ord order of derivative to penalize in baseline analysis
max.iter convergence criterion in baseline calculation
tol convergence criterion
sm.div smoothness divisor in baseline calculation
sm.norm.by method for smoothness penalty in baseline analysis
neg.div negativity divisor in baseline calculation
neg.norm.by method for negativity penalty in baseline analysis
rel.conv.crit logical; whether convergence criterion should be relative to size of current baseline estimate
zero.rm logical; whether to replace zeros with average of surrounding values
halve.search logical; whether to use a halving-line search if step leads to smaller value of function

Details

Goes through the entire directory raw.dir file-by-file and computes each baseline using baseline, then writes the spectrum and the baseline to a file in directory base.dir. The name of the new file is the same as the name of the old file with “.txt” replaced by “.RData”, and the new file is ready to be used by run.peaks.

The files in raw.dir must be in a specific format (future versions of the package will allow for more flexibility). The files should be two-column text files with mass in the first column and spectrum intensity in the second column. There should be no header row (just start the file with the first data point). The columns can be either comma-separated or whitespace-separated and the program will automatically detect which each file is. The decimal separator should be “,” as using “.” will cause errors in reading the files.

See baseline for details of all the parameters after par.file.

Value

No value returned; the files are simply created.

Note

If use.par.file == TRUE and other parameters are entered into the function call, then the parameters entered in the function call overwrite those read in from the file. Note that this is opposite from the behavior for FTICRMS versions 0.7 and earlier.

The values of sm.norm.by and neg.norm.by can be abbreviated and both have default value “baseline”.

Author(s)

Don Barkauskas (<barkda@wald.ucdavis.edu>)
**References**


**See Also**

baseline, run.peaks

---

**run.cluster.matrix**

*Identify Equivalent Peaks from Different Subjects*

**Description**

Takes the file generated by `run.lrg.peaks`, identifies equivalent peaks in each spectrum, and fills in missing values.

**Usage**

```r
run.cluster.matrix(pre.align = FALSE, align.method = c("PL", "spline", "affine", "none"), align.fcn = NA, trans.method = c("shiftedlog", "glog", "none"), add.par = 0, subtract.base = FALSE, lrg.only = TRUE, calc.all.peaks = FALSE, masses = NA, isotope.dist = 7, cluster.method = c("ppm", "constant", "usewidth"), cluster.constant = 10, num.pts = 5, R2.thresh = 0.98, oneside.min = 1, min.spect = 1, peak.method = c("parabola", "locmaxes"), bhbysubj = TRUE, covariates, root.dir = ".", base.dir, peak.dir, lrg.dir, lrg.file = "lrg_peaks.RData", overwrite = FALSE, use.par.file = FALSE, par.file = "parameters.RData")
```

**Arguments**

- `pre.align` either FALSE, or a numeric vector of shifts to apply to spectra, or a four-component list (of the form described in the Note section below) to be used before identifying peaks from different spectra
- `align.method` alignment algorithm for peaks
- `align.fcn` function (and inverse) to apply to masses before (and after) applying `align.method`; see below
trans.method  type of transformation to use on spectra before statistical analysis
add.par      additive parameter for "shiftedlog" or "glog" options for trans.method
subtract.base logical; whether to subtract calculated baseline from spectrum
lrg.only     logical; whether to consider only peaks that have at least one “large” peak; i.e.,
              identified by run.lrg.peaks
calc.all.peaks logical; whether to calculate all possible peaks or only sufficiently large ones
masses        specific masses to test
isotope.dist  maximum distance for declaring isotopes
cluster.method method for determining when two peaks from different spectra are the same
cluster.constant parameter used in running cluster.method
num.pts       number of consecutive points needed for peak fitting
R2.thresh     $R^2$ value needed for peak fitting
oneside.min   minimum number of points on each side of local maximum for peak fitting
min.spect     minimum number of spectra necessary for peak to be used in run.analysis
peak.method   method for locating peaks
bhbysubj      logical; whether to look for number of large peaks by subject (i.e., combining
              replicates) or by spectrum
covariates    data frame with rownames given by raw data files with extensions (e.g., “.txt”) stripped; only needed if bbhysubj == TRUE
root.dir      directory for parameters file and raw data;base.dir     directory for baseline files; default is paste(root.dir, "/Baselines", sep = "")
peak.dir      directory for peak location files; default is paste(root.dir, "/All_Peaks", sep = "")
lrg.dir       directory for large peaks file; default is paste(root.dir, "/Large_Peaks", sep = "")
lrg.file      name of file to store large peaks in
overwrite     logical; whether to replace existing files with new ones
use.par.file  logical; if TRUE, then parameters are read from par.file in directory root.dir
par.file      string containing name of parameters file

Details
Reads in information from file created by run.strong.peaks, calculates the cluster matrix, fills in missing values, and overwrites the file named lrg.file in lrg.dir. The resulting file contains variables

amps      data frame of amplitudes created by run.strong.peaks
centers    data frame of centers created by run.strong.peaks
clust.mat  data frame with columns given by samples and rows given by the distinct peaks in the samples
lrg.mat    data frame of same size as clust.mat with entries given by TRUE if the peak was large in that spectrum and FALSE
lrg.peaks  the data frame of significant peaks created by run.lrg.peaks
num.lrg    number of subjects (or spectra if bbhysubj == TRUE) with a large peak at the corresponding mass
and is ready to be used by `run.analysis`.

**Value**

No value returned; the file is simply created.

**Note**

If `use.par.file` = TRUE and other parameters are entered into the function call, then the parameters entered in the function call overwrite those read in from the file. Note that this is opposite from the behavior for FTICRMS versions 0.7 and earlier.

`align.method`, `cluster.method`, `peak.method`, and `trans.method` can be abbreviated.

If `align.fcn` is not NA, then it should consist of a list with components `fcn` and `inv`, each of class function. `align.fcn$fcn` should take a vector of masses as its argument and return a vector of transformed masses. (Typically, this will be transforming masses to frequencies; see Zhang (2005).) `align.fcn$inv` should be the inverse function of `align.fcn$fcn`.

If `align.method` = "spline", then alignment consists of making the transformed masses of the strong peaks all agree exactly with their means, then shifting the rest of the transformed masses via an interpolation spline generated using `interpSpline`. If `align.method` = "PL", then the same is done but interpolation is done piecewise linearly between the strong peaks. If `align.method` = "leastsq", then the transformed masses of the strong peaks are aligned to their means using a least-squares affine fit for each spectrum. In any of these cases, if there are no strong peaks, `align.method` is changed to "none" with a warning. If there is exactly one strong peak, then alignment is by a simple shift in each spectrum on the transformed masses. If there are exactly two strong peaks, then the alignment is by a simple affine transformation on the transformed masses in each spectrum. If `align.method` = "spline" and there are exactly three strong peaks, then alignment is piecewise affine on the transformed masses (i.e., identical to `align.method` = "PL").

If `align.method` = "leastsq", it is strongly recommended that you supply a value for `align.fcn` that makes the data points (approximately) equally-spaced.

Defining a value for `min.spect` can vastly speed up the run time at the (small) cost of a little flexibility in doing the statistical analysis in `run.analysis`. For exploratory data analysis, this should probably be left alone, but once the peak criterion has been established, further analyses will go much more quickly with `min.spect` re-defined. The value can either be an integer, which is interpreted as the number of spectra; or a number between 0 and 1, in which case it is interpreted as a fraction of the total number of spectra. In either case, the values of `clust.mat`, `lrg.mat`, and `num.lrg` saved in `lrg.file` are only those masses which have at least `min.spect` large peaks among the spectra.

`pre.align = FALSE` is used if the spectra have already been aligned by the mass spectrosocists. If it is not FALSE, it can either be a vector of additive shifts to be applied to the spectra, or a list with components `targets`, `actual`, and `align.method`. In the last case, `targets` is a vector of target masses, and `actual` is a matrix with `length(targets)` columns and a row for each spectrum, `actual[i,j]` being the mass in spectrum `i` that should be matched exactly to `target[j]`, with NA being a valid entry in `actual`. The alignment is then done as in the description in the above paragraph, depending on the number of non-missing values in row `i`.

Suppose `cluster.constant = K` and we have two peaks in different spectra with masses $m_1 < m_2$. If `cluster.method` = "constant", then the peaks are considered to be the same peak if we have $m_2 - m_1 < K$. If `cluster.method` = "ppm", then the peaks are considered to be the
same peak if we have \( m_2 - m_1 < K m_2 / 10^6 \). If \( \text{cluster.method} == \text{"usewidth"} \), then the algorithm uses the observation that \( \log(\text{Width.hat}) \) and \( \log(\text{Center.hat}) \) appear to be linearly related. Tolerances are computed using this relationship.

**Author(s)**

Don Barkauskas (<barkda@wald.ucdavis.edu>)

**References**


**See Also**

`run.lrg.peaks, run.strong.peaks, interpSpline`

---

**run.lrg.peaks**

*Extract "Large" Peaks from Files*

**Description**

Takes the files output by `run.peaks`, extracts “large” peaks, combines them into a single data frame, and writes the data frame to a file.

**Usage**

```r
run.lrg.peaks(trans.method = c("shiftedlog", "glog", "none"),
          add.par = 0, subtract.base = FALSE,
          root.dir = ".", peak.dir, base.dir, lrg.dir,
          lrg.file = lrg_peaks.RData, overwrite = FALSE,
          use.par.file = FALSE, par.file = "parameters.RData",
          calc.all.peaks = FALSE, gengamma.quantiles = TRUE,
          peak.thresh = 3.798194, subs)
```
**run.lrg.peaks**

**Arguments**

- **transNmethod**: type of transformation to use on spectra before statistical analysis
- **addNpar**: additive parameter for "shiftedlog" or "glog" options for trans.method
- **subtractNbase**: logical; whether to subtract calculated baseline from spectrum
- **rootNdir**: directory for parameters file and raw data
- **peakNdir**: directory for peak location files; default is paste(root.dir, "/All_Peaks", sep = "")
- **baseNdir**: directory for baseline files; default is paste(root.dir, "/Baselines", sep = "")
- **lrgNdir**: directory for large peaks file; default is paste(root.dir, "/Large_Peaks", sep = "")
- **lrg.file**: name of file to store large peaks in
- **overwrite**: logical; whether to replace existing files with new ones
- **useNparNfile**: logical; if TRUE, then parameters are read from par.file in directory root.dir
- **parNfile**: string containing name of parameters file
- **calc.allNpeaks**: logical; whether to calculate all possible peaks or only sufficiently large ones
- **gengamma.quantiles**: logical; whether to use generalized gamma quantiles when calculating large peaks
- **peakNthresh**: threshold for declaring large peak; see below
- **subs**: subset of spectra to use for analysis; see below

**Details**

Reads in information from each file created by run.peaks, extracts peaks which are “large” (see below), and creates the file lrg.file in lrg.dir. The resulting file contains the data frame lrg.peaks, which has columns

- **Center_hat**: estimated mass of peak
- **Max_hat**: estimated intensity of peak
- **Width_hat**: estimated width of peak
- **File**: name of file the peak was extracted from, with “.peaks.RData” deleted

and is ready to be used by run.strong.peaks.

**Value**

No value returned; the file is simply created.

**Note**

If use.par.file == TRUE and other parameters are entered into the function call, then the parameters entered in the function call overwrite those read in from the file. This is opposite from the behavior for FTICRMS versions 0.7 and earlier. trans.method can be abbreviated.
If `gengamma.quantiles` == TRUE, then a peak is “large” if it is at least `peak.thresh` times as large as the estimated baseline at that point.

If `gengamma.quantiles` == FALSE, then a peak is “large” if it has zero weight in the data generated by `run.peaks` for the spectrum it comes from when using Tukey’s biweight with parameter $K = 1.5 \times \text{peak.thresh}$ to estimate center and scale.

If `subs` is not defined, the algorithm finds large peaks for all files in `peak.dir`. If it is defined, `subs` can be logical or numeric or character; if it is defined, then the algorithm finds large peaks for all entries in `subs` (character) or `list.files(peak.dir)[subs]` (logical or numeric).

Author(s)

Don Barkauskas (<barkda@wald.ucdavis.edu>)

References


See Also

`run.peaks`, `run.cluster.matrix`

---

run.peaks

**Locate Potential Peaks in FT-ICR MS Spectra**

---

**Description**

Takes baseline-corrected data and locates potential peaks in the spectra.

**Usage**

```r
run.peaks(trans.method = c("shiftedlog", "glog", "none"),
              add.par = 0, subtract.base = FALSE, root.dir = ".",
              base.dir, peak.dir, overwrite = FALSE,
              use.par.file = FALSE, par.file = "parameters.RData",
              num.pts = 5, R2.thresh = 0.98, oneside.min = 1,
              peak.method = c("parabola", "locmaxes"),
              calc.all.peaks = FALSE, gengamma.quantiles = TRUE,
              peak.thresh = 3.798194)
```
run.peaks

Arguments

- **trans.method**: type of transformation to use on spectra before statistical analysis
- **add.par**: additive parameter for "shiftedlog" or "glog" options for trans.method
- **subtract.base**: logical; whether to subtract calculated baseline from spectrum
- **root.dir**: directory for parameters file and raw data
- **base.dir**: directory for baseline files; default is paste(root.dir, "/Baselines", sep = "")
- **peak.dir**: directory for peak location files; default is paste(root.dir, "/All_Peaks", sep = "")
- **overwrite**: logical; whether to replace existing files with new ones
- **use.par.file**: logical; if TRUE, then parameters are read from par.file in directory root.dir
- **par.file**: string containing name of parameters file
- **num.pts**: number of consecutive points needed for peak fitting
- **R2.thresh**: \( R^2 \) value needed for peak fitting
- **oneside.min**: minimum number of points on each side of local maximum for peak fitting
- **peak.method**: method for locating peaks
- **calc.all.peaks**: logical; whether to calculate all possible peaks or only sufficiently large ones
- **gengamma.quantiles**: logical; whether to use generalized gamma quantiles when calculating large peaks
- **peak.thresh**: threshold for declaring large peak; see below

Details

Reads in information from each file created by run.baselines, calls locate.peaks to find potential peaks, and writes the output to a file in directory peak.dir. The name of each new file is the same as the name of the old file with "RData" replaced by "_peaks.RData". The resulting file contains the data frame all.peaks, which has columns

- **Center_hat**: estimated mass of peak
- **Max_hat**: estimated intensity of peak
- **Width_hat**: estimated width of peak

and is ready to be used by run.lrg.peaks.

The parameters gengamma.quantiles and peak.thresh are relevant only if calc.all.peaks = FALSE. In that case, if gengamma.quantiles == TRUE, then peak.thresh is interpreted as a multiplier for the baseline. Anything larger than peak.thresh times the estimated baseline is declared to be a real peak. If gengamma.quantiles == FALSE, then peak.thresh is interpreted as two-thirds of the value of \( K \) used in a Tukey’s biweight estimation of center and scale (so roughly equal to the number of standard deviations above the mean for iid normal data). Anything with weight zero in the calculation is then declared to be a real peak.
run.strong.peaks

Value

No value returned; the files are simply created.

Note

If use.par.file == TRUE and other parameters are entered into the function call, then the parameters entered in the function call overwrite those read in from the file. Note that this is opposite from the behavior for FTICRMS versions 0.7 and earlier.

peak.method and trans.method can be abbreviated.

Using calc.all.peaks == FALSE can speed up computation time immensely, but will affect the final result. It probably won’t affect it much, but caveat emptor.

Author(s)

Don Barkauskas (<barkda@wald.ucdavis.edu>)

References


See Also

run.baselines, run.lrg.peaks, locate.peaks

---

run.strong.peaks  Locate Peaks that are "Large" in All Samples

Description

Takes the file generated by run.peaks, extracts all peaks that are “large” in all samples, and writes the results to a file.

Usage

run.strong.peaks(cor.thresh = 0.8, isotope.dist = 7, pre.align = FALSE, align.method = c("PL", "spline", "affine", "none"), alignfcn = NA, root.dir = ".", lrg.dir, lrg.file = "lrg_peaks.RData", overwrite = FALSE, use.par.file = FALSE, par.file = "parameters.RData")
Arguments

cor.thresh threshold correlation for declaring isotopes
isotope.dist maximum distance for declaring isotopes
pre.align either FALSE, or a numeric vector of shifts to apply to spectra, or a three-component list (of the form described in the Note section below) to be used before identifying peaks from different spectra
align.method alignment algorithm for peaks
align.fcn function (and inverse) to apply to masses before (and after) applying align.method; see below
root.dir directory for parameters file and raw data
lrg.dir directory for large peaks file; default is `paste(root.dir, "/Large_Peaks", sep = "")`
lrg.file name of file to store large peaks in
overwrite logical; whether to replace existing files with new ones
use.par.file logical; if TRUE, then parameters are read from par.file in directory root.dir
par.file string containing name of parameters file

Details

Reads in information from file created by `run.lrg.peaks`, locates peaks which appear in all samples, and overwrites the file `lrg.file` in `lrg.dir`. The resulting file contains variables

- `amps` data frame of amplitudes of non-isotope peaks that occur in all samples
- `centers` data frame of centers of non-isotope peaks that occur in all samples
- `lrg.peaks` the data frame of significant peaks created by `run.lrg.peaks`

and is ready to be used by `run.cluster.matrix`.

Value

No value returned; the file is simply created.

Note

If `use.par.file` == TRUE and other parameters are entered into the function call, then the parameters entered in the function call overwrite those read in from the file. Note that this is opposite from the behavior for `FTICRMS` versions 0.7 and earlier.

If `align.fcn` is not NA, then it should consist of a list with components `fcn` and `inv`, each of class function. `align.fcn$fcn` should take a vector of masses as its argument and return a vector of transformed masses. (Typically, this will be transforming to the frequency domain; see Zhang (2005)). `align.fcn$inv` should be the inverse function of `align.fcn$fcn`. If `align.method` == "leastsq", it is strongly recommended that you supply a value for `align.fcn` that makes the masses (approximately) equally-spaced.
align.method can be abbreviated. If align.method == "spline", then alignment consists of making the transformed masses of the strong peaks all agree exactly with their means, then shifting the rest of the transformed masses via a cubic interpolation spline generated using interpSpline. If align.method == "PL", then the same is done but interpolation is piecewise linear between the strong peaks. If align.method == "leastsq", then the transformed masses of the strong peaks are aligned to their means using a least-squares affine fit for each spectrum. In any of these cases, if there are no strong peaks, align.method is changed to "none" with a warning. If there is exactly one strong peak, then alignment is by a simple shift in each spectrum on the transformed masses. If there are exactly two strong peaks, then the alignment is by a simple affine transformation on the transformed masses in each spectrum. If align.method == "spline" and there are exactly three strong peaks, then alignment is piecewise affine on the transformed masses (i.e., identical to using align.method = "PL").

pre.align = FALSE is used if the spectra have already been aligned by the mass spectroscopists. If it is not FALSE, it can either be a vector of additive shifts to be applied to the spectra, or a list with components targets, actual, and align.method. In the last case, targets is a vector of target masses, and actual is a matrix with length(targets) columns and a row for each spectrum, actual[i,j] being the mass in spectrum i that should be matched exactly to target[j], with NA being a valid entry in actual. The alignment is then done row-by-row as in the description in the above paragraph, depending on the number of non-missing values in row i).

Author(s)

Don Barkauskas (<barkda@wald.ucdavis.edu>)

References


See Also

run.lrg.peaks, run.cluster.matrix, interpSpline
Index

*Topic package
  FTICRMS-package, 2

anova, 5, 6
baseline, 2, 18, 19
display.tests, 5
eextract.pars, 6, 12, 15
formula, 10, 14
FTICRMS, 7, 10, 11, 16, 18, 21, 23, 26, 27
FTICRMS (FTICRMS-package), 2
FTICRMS-package, 2
interpspline, 21, 22, 28
lm, 6
locate.peaks, 9, 25, 26
make.par.file, 8, 10, 13, 14
run.all, 13
run.analysis, 5–8, 11–14, 18, 20, 21
run.baselines, 4, 11–14, 17, 25, 26
run.cluster.matrix, 11–15, 19, 24, 27, 28
run.lrg.peaks, 7, 11–14, 19, 20, 22, 25–28
run.peaks, 10–14, 18, 19, 22–24, 24, 26
run.strong.peaks, 7, 11, 13, 14, 17, 20, 22, 23, 26
summary, 5
t.test, 5, 6, 16