Package ‘hyperSpec’

November 19, 2016

Type Package

Title Work with Hyperspectral Data, i.e. Spectra + Meta Information
(Spatial, Time, Concentration, ...)

Version 0.98-20161118

Date 2016-11-18

Author Claudia Beleites

Maintainer Claudia Beleites <chemometrie@beleites.de>

Description Comfortable ways to work with hyperspectral data sets.
I.e. spatially or time-resolved spectra, or spectra with any other kind
of information associated with each of the spectra. The spectra can be data
as obtained in XRF, UV/VIS, Fluorescence, AES, NIR, IR, Raman, NMR, MS,
etc. More generally, any data that is recorded over a discretized variable,
e.g. absorbance = f (wavelength), stored as a vector of absorbance values
for discrete wavelengths is suitable.

License GPL (>= 3)

LazyLoad yes

LazyData yes

Depends R (>= 2.15), lattice, grid, ggplot2 (>= 2.2.0)

Suggests R.matlab, tripack, deldir, rgl, plotrix, testthat, sp,
baseline, compiler, inline, Rcpp, MASS, fastcluster, pls,
mvtnorm, digest

Imports methods, utils, latticeExtra, svUnit

URL http://hyperSpec.r-forge.r-project.org/

Collate 'validate.R' 'hyperspec-class.R' 'barbiturates.R'
'paracetamol.R' 'laser.R' 'chondro.R' 'flu.R'
'hyperspec-package.R' 'paste.row.R' 'Arith.R' 'Compare.R'
'DollarNames.R' 'Math.R' 'Summary.R' 'aggregate.R'
'all.equal.R' 'apply.R' 'as.data.frame.R' 'bind.R'
'call.list.R' 'chk.hy.R' 'colMeans.R' 'collapse.R' 'cov.R'
'decomposition.R' 'dim.R' 'dimnames.R' 'empty.R' 'w12i.R'
'extract.R' 'factor2num.R' 'fileio.optional.R' 'getbynames.R'

1
R topics documented:

'initialize.R' 'labels.R' 'plotmap.R' 'levelplot.R'
'makeraster.R' 'map.identify.R' 'map.sel.poly.R'
'mark.dendrogram.R' 'mark.peak.R' 'matplotlib.palette.R'
'mean_sd.R' 'merge.R' 'mvtnorm.R' 'normalize01.R' 'options.R'
'orderwl.R' 'pearson.dist.R' 'plot.R' 'plot.R' 'plotmat.R'
'plotspc.R' 'plotvoronoi.R' 'qplot.R' 'qplotmixmap.R'
'quantile.R' 'rbind.fill.R' 'read.ENVI.Nicolet.R' 'read.ENVI.R'
'read.ini.R' 'read.jdx.R' 'read.jdx.Shimadzu.R'
'read.mat.Cytospec.R' 'read.mat.Witec.R' 'read.spc.Kaiser.R'
'read.spc.R' 'read.spe.R' 'read.txt.Horiba.R'
'read.txt.Shimadzu.R' 'read.txt.long.R' 'read.txt.wide.R'
'replace.R' 'sample.R' 'scale.R' 'scan.txt.Witec.R'
'scan.asc.Andor.R' 'scan.txt.Renishaw.R' 'seq.R' 'show.R'
'spc.NA.approx.R' 'spc.bin.R' 'spc.fit.poly.R' 'spc.identify.R'
'spc.loess.R' 'spc.rubberband.R' 'spc.spline.R' 'split.R'
'split.string.R' 'splitdots.R' 'subset.R' 'sweep.R'
'trellis.factor.key.R' 'units.R' 'unittest.R' 'vandermonde.R'
'wc.R' 'wl.R' 'wleval.R' 'write.txt.long.R' 'write.txt.wide.R'
'y-pastenames.R' 'zzz.R'

---

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-11-19 10:14:29

---

R topics documented:

hyperSpec-package .................................................. 4
.DollarNames.hyperSpec ........................................ 5
aggregate .......................................................... 5
apply ................................................................. 7
Arith ................................................................. 9
as.character.hyperSpec-method .................................... 11
as.data.frame .................................................... 12
barbiturates ....................................................... 14
bind ................................................................. 15
chk.hy ............................................................. 17
chondro ............................................................. 18
collapse ........................................................... 19
colSums ............................................................ 20
Comparison ......................................................... 21
cov.hyperSpec,missing-method ................................... 23
decomposition ..................................................... 24
dimnames.hyperSpec-method ....................................... 25
empty ............................................................... 27
flu ................................................................. 27
hy.getOptions ................................................... 28
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>hy.unittest</td>
<td>29</td>
</tr>
<tr>
<td>hyperSpec-class</td>
<td>30</td>
</tr>
<tr>
<td>initialize</td>
<td>31</td>
</tr>
<tr>
<td>labels&lt;-</td>
<td>32</td>
</tr>
<tr>
<td>laser</td>
<td>33</td>
</tr>
<tr>
<td>legendright</td>
<td>34</td>
</tr>
<tr>
<td>makeraster</td>
<td>36</td>
</tr>
<tr>
<td>map.sel.poly</td>
<td>37</td>
</tr>
<tr>
<td>mark.dendrogram</td>
<td>39</td>
</tr>
<tr>
<td>markpeak</td>
<td>40</td>
</tr>
<tr>
<td>Math2,hyperSpec-method</td>
<td>41</td>
</tr>
<tr>
<td>matlab.palette</td>
<td>42</td>
</tr>
<tr>
<td>mean_sd,numeric-method</td>
<td>43</td>
</tr>
<tr>
<td>merge,hyperSpec,hyperSpec-method</td>
<td>45</td>
</tr>
<tr>
<td>ncol,hyperSpec-method</td>
<td>46</td>
</tr>
<tr>
<td>normalize01</td>
<td>47</td>
</tr>
<tr>
<td>orderwl</td>
<td>48</td>
</tr>
<tr>
<td>paracetamol</td>
<td>49</td>
</tr>
<tr>
<td>pearson.dist</td>
<td>50</td>
</tr>
<tr>
<td>plot-methods</td>
<td>51</td>
</tr>
<tr>
<td>plotc</td>
<td>52</td>
</tr>
<tr>
<td>plotmap</td>
<td>54</td>
</tr>
<tr>
<td>plotmat</td>
<td>57</td>
</tr>
<tr>
<td>plotspc</td>
<td>58</td>
</tr>
<tr>
<td>qplotc</td>
<td>61</td>
</tr>
<tr>
<td>qplotmap</td>
<td>63</td>
</tr>
<tr>
<td>qplotmixmap</td>
<td>64</td>
</tr>
<tr>
<td>qplotspc</td>
<td>65</td>
</tr>
<tr>
<td>rbind.fill.matrix</td>
<td>66</td>
</tr>
<tr>
<td>read.ENVI.Nicolet</td>
<td>67</td>
</tr>
<tr>
<td>read.ini</td>
<td>70</td>
</tr>
<tr>
<td>read.jdx</td>
<td>71</td>
</tr>
<tr>
<td>read.mat.Cytospec</td>
<td>72</td>
</tr>
<tr>
<td>read.spc</td>
<td>73</td>
</tr>
<tr>
<td>read.spc.Kaiser</td>
<td>74</td>
</tr>
<tr>
<td>read.spe</td>
<td>75</td>
</tr>
<tr>
<td>read.txt.Horiba</td>
<td>77</td>
</tr>
<tr>
<td>read.txt.long</td>
<td>78</td>
</tr>
<tr>
<td>read.txt.Shimadzu</td>
<td>81</td>
</tr>
<tr>
<td>rmmvnorm</td>
<td>81</td>
</tr>
<tr>
<td>sample,hyperSpec-method</td>
<td>83</td>
</tr>
<tr>
<td>scale,hyperSpec-method</td>
<td>84</td>
</tr>
<tr>
<td>scan.asc.Andor</td>
<td>85</td>
</tr>
<tr>
<td>scan.txt.Renishaw</td>
<td>86</td>
</tr>
<tr>
<td>scan.txt.Witec</td>
<td>88</td>
</tr>
<tr>
<td>seq.hyperSpec</td>
<td>89</td>
</tr>
<tr>
<td>spc.bin</td>
<td>90</td>
</tr>
<tr>
<td>spc.fit.poly</td>
<td>91</td>
</tr>
</tbody>
</table>
Package hyperSpec

Description

Interface for hyperspectral data sets This package gives an interface to handle hyperspectral data sets in R. Hyperspectral data are spatially or time-resolved spectra, or spectra with any other kind of information associated with the spectra. E.g. spectral maps or images, time series, calibration series, etc.

Details

The spectra can be data as obtained in XRF, UV/VIS, Fluorescence, AES, NIR, IR, Raman, NMR, MS, etc.

More generally, any data that is recorded over a discretized variable, e.g. absorbance = f (wavelength), stored as a vector of absorbance values for discrete wavelengths is suitable.

Author(s)

C. Beleites
Maintainer: Claudia Beleites <claudia.beleites@chemometrix.eu>

See Also

citation ("hyperSpec") produces the correct citation.
package?hyperSpec for information about the package
class?hyperSpec for details on the S4 class provided by this package.
command line completion for $

Usage

## S3 method for class 'hyperSpec'
.DollarNames(x, pattern)

Arguments

- **x**: the hyperSpec object
- **pattern**: pattern to look for

Value

the name of the extra data slot

Author(s)

C. Beleites

See Also

.DollarNames

aggregate

aggregate hyperSpec objects

Description

Compute summary statistics for subsets of a hyperSpec object.

Usage

## S4 method for signature 'hyperSpec'
aggregate(x, by = stop("by is needed"),
  FUN = stop("FUN is needed."), ..., out.rows = NULL, append.rows = NULL,
  by.isindex = FALSE)
Arguments

x a hyperSpec object
by grouping for the rows of x@data.
Either a list containing an index vector for each of the subgroups or a vector that
can be split in such a list.
FUN function to compute the summary statistics
... further arguments passed to FUN
out.rows number of rows in the resulting hyperSpec object, for memory preallocation.
append.rows If more rows are needed, how many should be appended?
Defaults to 100 or an estimate based on the percentage of groups that are still to
be done, whatever is larger.
by.isindex If a list is given in by: does the list already contain the row indices of the groups?
If FALSE, the list in by is computed first (as in aggregate).

Details
aggregate applies FUN to each of the subgroups given by by. It combines the functionality of
aggregate, tapply, and ave for hyperSpec objects.
aggregate avoids splitting x@data.
FUN does not need to return exactly one value. The number of returned values needs to be the same
for all wavelengths (otherwise the result could not be a matrix), see the examples.
If the initially preallocated data.frame turns out to be too small, more rows are appended and a
warning is issued.

Value
A hyperSpec object with an additional column @data$. aggregate tracing which group the rows
belong to.

Author(s)
C. Beleites

See Also
tapply, aggregate, ave

Examples
cluster.means <- aggregate (chondro, chondro$clusters, mean_pm_sd)
plot(cluster.means, stacked = ".aggregate", fill = ".aggregate",
col = matlab.dark.palette (3))

### make some "spectra"
spc <- new ("hyperSpec", spc = sweep (matrix (rnorm (10*20), ncol = 20), 1, (1:10)*5, "+")

### 3 groups
apply

color <- c("red", "blue", "black")
by <- as.factor(c(1, 1, 1, 1, 1, 5, 1, 2, 2))
by
plot (spc, "spc", col = color[by])

## Example 1: plot the mean of the groups
plot (aggregate (spc, by, mean), "spc", col = color, add = TRUE,
     lines.args = list(lwd = 3, lty = 2))

## Example 2: FUN may return more than one value (here: 3)
plot (aggregate (spc, by, mean_pm_sd), "spc",
     col = rep(color, each = 3), lines.args = list(lwd = 3, lty = 2))

## Example 3: aggregate even takes FUN that return different numbers of
## values for different groups
plot (spc, "spc", col = color[by])

weird.function <- function (x){
  if (length (x) == 1)
    x + 1 : 10
  else if (length (x) == 2)
    NULL
  else
    x [1]
}

agg <- aggregate (spc, by, weird.function)
agg$.aggregate
plot (agg, "spc", add = TRUE, col = color[agg$.aggregate],
     lines.args = list (lwd = 3, lty = 2))
apply

MARGIN  The subscript which the function will be applied over.
1 indicates rows (FUN is applied to each spectrum),
2 indicates columns (FUN is applied to each wavelength),
1 : 2 indicates that FUN should be applied to each single element of the spectra
matrix. Note that many basic mathematical functions are already defined for
hyperSpec objects (see Math).
If MARGIN is missing, the whole spectra matrix is handed to FUN, see also the
examples.

FUN  function to compute the summary statistics
...  further arguments passed to FUN
label.wl, label.spc  new labels for wavelength and spectral intensity axes
new.wavelength  for MARGIN = 2: numeric vector or name of the argument in ...that is to be used
(character) as wavelength axis of the resulting object.

Details
The generic functions of group Math are not defined for hyperSpec objects. Instead, apply can be
used. For functions like log that work on scalars, MARGIN = 1 : 2 gives the appropriate behaviour.
spcapply does the same as apply with MARGIN = 1, but additionally allows to set a new wavelength
axis and adjust the labels.
wlapply does the same as apply with MARGIN = 2, but additionally allows to set a new wavelength
axis and adjust the labels.

Value
A hyperSpec object

Author(s)
C. Beleites

See Also
apply, for applying FUN to subgroups of the hyperSpec object: aggregate.

Examples

```
plotspc (apply (chondro, 2, range))

avgflu <- apply (flu, 1, mean,
   label.spc = expression (bar (I)),
   new.wavelength = mean (wl (flu)))

avgflu

flu[[,405:407]]
```
apply (flu, 1:2, "*", -1)[[,,405:407]]

## without MARGIN the whole matrix is handed to FUN
apply (flu [,405:407], , print) [[]]

## whereas MARGIN = 1 : 2 leads to FUN being called for each element separately
apply (flu [,405:407], 1 : 2, print) [[]]

---

**Arith**

**Arithmetrical Operators for hyperSpec objects**

**Description**

Arithmetical Operators: +, -, *, /, ^, %%, %/%, %*% for hyperSpec objects

**Usage**

### S4 method for signature 'hyperSpec,hyperSpec'

Arith(e1, e2)

### S4 method for signature 'hyperSpec,numeric'

Arith(e1, e2)

### S4 method for signature 'hyperSpec,matrix'

Arith(e1, e2)

### S4 method for signature 'hyperSpec,missing'

Arith(e1, e2)

### S4 method for signature 'numeric,hyperSpec'

Arith(e1, e2)

### S4 method for signature 'matrix,hyperSpec'

Arith(e1, e2)

### S4 method for signature 'hyperSpec,hyperSpec'

x %*% y

### S4 method for signature 'hyperSpec,matrix'

x %*% y

### S4 method for signature 'matrix,hyperSpec'

x %*% y
Arguments

e1, e2 or
x, y either two hyperSpec objects or
one hyperSpec object and matrix of same size as hyperSpec[[[]]] or
a vector which length equalling either the number of rows or the number of
wavelengths of the hyperSpec object, or
a scalar (numeric of length 1).

Details

The arithmetical operators +, -, *, /, ^, %%/%, and %*% for hyperSpec objects.

You can use these operators in different ways:

e1 + e2
'+' (e1, e2)

x %*% y
'%*%'(x, y)

-x

The arithmetical operators +, -, *, /, ^, %%/%, and %*% work on the spectra matrix of the
hyperSpec object. They have their usual meaning (see Arithmetic). The operators work also
with one hyperSpec object and a numeric object or a matrices of the same size as the spectra
matrix of the hyperSpec object.

With numeric vectors sweep is most probably more appropriate.

If you want to calculate on the extra data as well, use the data.frame hyperSpec$data directly or
as.data.frame (x).

Value

hyperSpec object with the new spectra matrix.

Author(s)

C. Beleites

See Also

sweep-methods for calculations involving a vector and the spectral matrix.
S4groupGeneric for group generic methods.
Arithmetic for the base arithmetic functions.
Comparison for comparison operators, Math for mathematical group generic functions (Math and
Math2 groups) working on hyperSpec objects.
matmult for matrix multiplications with %*%.
Examples

```r
flu + flu
1 / flu
all((flu + flu - 2 * flu)[[1]] == 0)
-flu
flu / flu$c
```

---

**as.character,hyperSpec-method**

*Convert a hyperSpec object to character strings for Display print, show, and summary show the result of as.character.*

---

**Description**

print, show, and summary differ only in the defaults. show displays the range of values instead, print shows the overview giving the first and last values of each data column (fastest), summary displays the logbook in addition.

**Usage**

```r
## S4 method for signature 'hyperSpec'
as.character(x, digits = getOption("digits"),
    range = TRUE, max.print = 5, shorten.to = c(2, 1))

## S4 method for signature 'hyperSpec'
show(object)

## S4 method for signature 'hyperSpec'
print(x, range = FALSE, ...)

## S4 method for signature 'hyperSpec'
summary(object, ...)
```

**Arguments**

- `x` : a hyperSpec object
- `digits` : number of digits handed over to format
- `range` : should the values be indicated as range rather then first and last elements?
- `max.print` : maximum number of elements to be printed (of a variable)
- `shorten.to` : if a vector is longer than max.print, only the first shorten.to[1] and the last shorten.to[2] elements are printed
- `object` : a hyperSpec object
- `...` : print and summary hand further arguments to as.character
as.data.frame

Value

as.character returns a character vector fit to be printed by `cat` with `sep = "\n"`

print invisibly returns x after printing, show returns an invisible NULL.

See Also

as.character
show
print
summary

Examples

chondro

show (chondro)

summary (chondro)

print (chondro, range = TRUE)

as.data.frame

Conversion of a hyperSpec object into a data.frame or matrix

as.data.frame returns x@data (as.data.frame) as.matrix returns the spectra matrix x@data$spc as matrix

Description

Conversion of a hyperSpec object into a data.frame or matrix as.data.frame returns x@data (as data.frame) as.matrix returns the spectra matrix x@data$spc as matrix

as.wide.df converts the spectra matrix to a data.frame. The extra data together with this data is returned. The column names of the spectra matrix are retained (if they are numbers, without preceding letters).

The data.frame returned by as.long.df is guaranteed to have columns spc and .wavelength. If nwl (x) == 0 these columns will be NA.

as.t.df produces a 'transposed' data.frame with columns containing the spectra.
Usage

```r
## S3 method for class 'hyperSpec'
as.data.frame(x, row.names = TRUE, optional = NULL, ...)

## S3 method for class 'hyperSpec'
as.matrix(x, ...)

as.wide.df(x)

as.long.df(x, rownames = FALSE, wl.factor = FALSE, na.rm = TRUE)

as.t.df(x)
```

Arguments

- **x**: a `hyperSpec` object
- **row.names**: if `TRUE`, a column `.row` is created containing row names or row indices if no rownames are set. If character vector, the rownames are set accordingly.
- **optional**: ignored
- **...**: ignored
- **rownames**: should the rownames be in column `.rownames` of the long-format data.frame?
- **wl.factor**: should the wavelengths be returned as a factor (instead of numeric)?
- **na.rm**: if `TRUE`, rows where spc is not `NA` are deleted.

Value

- `x@data` and `x@data$spc (== x$spc == x [[]])`, respectively.
- `as.wide.df` returns a data.frame that consists of the extra data and the spectra matrix converted to a data.frame. The spectra matrix is expanded *in place*.
- `as.long.df` returns the stacked or molten version of `x@data`. The wavelengths are in column `.wavelength`.
- `as.t.df` returns a data.frame similar to `as.long.df`, but each spectrum in its own column. This is useful for exporting summary spectra, see the example.

Author(s)

C. Beleites

See Also

- `as.data.frame`
- `as.matrix`
  - for a shortcut to `as.matrix`
- `stack` and `melt` or `melt` for other functions producing long-format data.frames.
Examples

```r
as.data.frame (chondro [1:3, 600 - 620])
as.matrix (chondro [1:3, 600 - 620])
lm (c ~ spc, data = flu [,,450])
as.wide.df (chondro [1:5, 600 - 610])
summary (as.wide.df (chondro [1:5, 600 - 610]))
as.long.df (flu [,, 405 - 410])
summary (as.long.df (flu [,, 405 - 410]))
summary (as.long.df (flu [,, 405 - 410], rownames = TRUE))
summary (as.long.df (flu [,, 405 - 410], wl.factor = TRUE))
df <- as.t.df (apply (chondro, 2, mean_pm_sd))
head (df)
if (require (ggplot2)){
  ggplot (df, aes (x = .wavelength)) +
  geom_ribbon (aes (ymin = mean.minus.sd, ymax = mean.plus.sd),
               fill = "#00000040") +
  geom_line (aes (y = mean))
}
```

barbiturates  
Barbiturates Spectra from .spc example files A time series of mass spectra in a list of hyperSpec objects.

Description

Barbiturates Spectra from .spc example files A time series of mass spectra in a list of hyperSpec objects.

Format

The data sets consists of 286 spectra. They are the result of importing the BARBITUATES.SPC example data from Thermo Galactic’s spc file format specification.

Author(s)

C. Beleites and Thermo Galactic

References

The raw data is available at [http://hyperspec.r-forge.r-project.org/blob/fileio.zip](http://hyperspec.r-forge.r-project.org/blob/fileio.zip)
Examples

barbiturates [1:3]
length (barbiturates)

barb <- collapse (barbiturates)
barb <- orderwl (barb)

plot (barb [1:3], lines.args = list (type = "h"),
     col = matlab.dark.palette (3), stacked = TRUE,
     stacked.args = list (add.factor = .2))

if (require (latticeExtra)){
  levelplot (spc ~ wavelength * z, log (barb), panel = panel.levelplot.points,
             cex = 0.3, col = "#0000000", col.regions = matlab.palette (20))
}

plotc (apply (barb [, , 42.9:43.2], 1, sum, na.rm = TRUE), spc ~ z,
       panel = panel.lines, ylab = expression (I[m/z == 43] / "a.u."))

bind	Binding hyperSpec Objects Two S3 functions cbind.hyperSpec and
rbind.hyperSpec act as an interfaces to cbind2 and rbind2 because
neither rBind and cBind nor S4 versions of cbind and rbind do work
at the moment.

Description

While it is now possible to do S4 despatch on ...(), defining such S4 methods for cbind and
rbind breaks the binding of Matrix objects. Therefore, two S3 methods rbind.hyperSpec and
cbind.hyperSpec are defined.
cbind2 binds the spectral matrices of two hyperSpec objects by column. All columns besides spc
with the same name in x@data and y@data must have the same elements. Rows are ordered before
checking.

rbind2 binds two hyperSpec objects by row. They need to have the same columns.

Usage

bind (direction = stop("direction ('c' or 'r') required"), ...,
      wl.tolerance = hy.getOption("wl.tolerance"))

cbind.hyperSpec(...)

rbind.hyperSpec(...)

## S4 method for signature 'hyperSpec,hyperSpec'

cbind2(x, y)
## Arguments

- **direction**
  - "r" or "c" to bind rows or columns
- **...**
  - The hyperSpec objects to be combined.
  - Alternatively, one list of hyperSpec objects can be given to bind.
- **wl.tolerance**
  - `rbind` and `rbind2` check for equal wavelengths with this tolerance.
- **x, y**
  - hyperSpec objects

## Details

`bind` does the common work for both column- and row-wise binding.

## Value

A hyperSpec object, possibly with different row order (for `bind ("c", ...{})` and `cbind2`).

## Note

You might have to make sure that the objects either all have or all do not have rownames and/or colnames.

## Author(s)

C. Beleites

## See Also

- `rBind`, `cBind`, `rbind2`, `cbind2`, `rbind`, `cbind`, `merge` and `collapse` for combining objects that do not share spectra or wavelengths, respectively.

## Examples

```r
chondro
bind ("r", chondro, chondro)
rbind (chondro, chondro)
ccbind (chondro, chondro)
bind ("r", list (chondro, chondro, chondro))
```


\[ x \leftarrow \text{chondro}[,, \ 600:605] \]
\[ x@a \leftarrow 1 \]
\[ x@data \leftarrow x@data[, \ \text{sample} (\text{ncol} (x), \text{ncol} (x))] \] # reorder columns

\[ y \leftarrow \text{chondro} [\text{nrow} (\text{chondro}) \ : \ 1, \ 1730:1750] \] # reorder rows
\[ y@b \leftarrow 2 \]
\[ \text{cbind2} (x, y) \] # works

\[ y@y[3] \leftarrow 5 \]
\[ \text{try} (\text{cbind2} (x, y)) \] # error

chk.hydro Validation of hyperSpec objects

Description
Check whether an object is a hyperSpec object and validate the object

Usage
chk.hydro(object)

Arguments
object the object to check

Value
TRUE if the check passes, otherwise stop with an error.

Author(s)
C. Beleites

See Also
validObject

Examples
chk.hydro (chondro)
validObject (chondro)
chondro

Raman spectra of 2 Chondrocytes in Cartilage A Raman-map (laterally resolved Raman spectra) of chondrocytes in cartilage.

Description
See the vignette vignette ("chondro", package = "hyphenSpec").

Format
The data set has 875 Raman spectra measured on a 25 × 35 grid with 1 micron step size. Spatial information is in chondro$x and chondro$y. Each spectrum has 300 data points in the range of ca. 600 - 1800 cm⁻¹.

Author(s)
A. Bonifacio and C. Beleites

References
The raw data is available at http://hyperspec.r-forge.r-project.org/blob/chondro.zip

Examples

```r
chondro

## do baseline correction
baselines <- spc.fit.poly.below (chondro)
chondro <- chondro - baselines

## area normalization
chondro <- chondro / colMeans (chondro)

## subtract common composition
chondro <- chondro - quantile (chondro, 0.05)

cols <- c ("dark blue", "orange", "#C02020")
plotmap (chondro, clusters ~ x * y, col.regions = cols)

cluster.means <- aggregate (chondro, chondro$clusters, mean_pm_sd)
plot (cluster.means, stacked = ".aggregate", fill = ".aggregate", col = cols)

## plot nucleic acid bands
plotmap (chondro[, , c(728, 782, 1098, 1240, 1482, 1577)],
         col.regions = colorRampPalette (c ("white", "gold", "dark green"), space = "Lab") (20))
```
collapse

Description

collapse/bind several hyperSpec objects into one object

Usage

collapse(..., wl.tolerance = hy.getOption("wl.tolerance"))

Arguments

... hyperSpec objects to be collapsed into one object. Instead of giving several arguments, a list with all objects to be collapsed may be given.
wl.tolerance tolerance to decide which wavelengths are considered equal.

Details

The spectra from all objects will be put into one object. The resulting object has all wavelengths that occur in any of the input objects, \texttt{wl.tolerance} is used to determine which difference in the wavelengths is tolerated as equal. The returned object has wavelengths rounded according to the precision indicated by \texttt{wl.tolerance}.

Data points corresponding to wavelengths not in the original spectrum will be set to NA. Extra data is combined in the same manner.

If the objects are named, the names will be preserved in extra data column \$.name.

Value

a hyperSpec object

Author(s)

C. Beleites

See Also

merge to merge hyperSpec objects that share wavelengths but contain different spectra, rbind, and rbind.fill for

Examples

barbiturates [1:3]
barb <- collapse (barbiturates [1:3])
barb

a <- barbiturates [[1]]
b <- barbiturates [[2]]
c <- barbiturates [[3]]

a
b
c
collapse (a, b, c)

---

## colSums

**colSums, colMeans, rowSums and rowMeans functions for hyperSpec objects**

---

### Description

hyperSpec objects can use the base functions *colMeans, colSums, rowMeans* and *rowSums*.

### Usage

```
## S4 method for signature 'hyperSpec'
colMeans(x, na.rm = TRUE, ..., label.spc)

## S4 method for signature 'hyperSpec'
colSums(x, na.rm = TRUE, ..., label.spc)

## S4 method for signature 'hyperSpec'
rowMeans(x, na.rm = TRUE, ..., label.wavelength)

## S4 method for signature 'hyperSpec'
rowSums(x, na.rm = TRUE, ..., label.wavelength)
```

### Arguments

- `x` hyperSpec object
- `na.rm, ...` further parameters to the base functions
  - `na.rm` defaults to `TRUE` for hyperSpec objects.
- `label.spc` labels for the intensity axis for loadings-like (col) statistics
- `label.wavelength` labels for the wavelength axis for scores-like (row) statistics

### See Also

- `colSums`
Comparison

**Examples**

```r
colMeans (flu)
colSums (flu)
colSums (flu)
rowSums (flu)
```

**Comparison of hyperSpec objects**

**Description**

The comparison operators `>`, `<`, `>=`, `<=`, and `!=` for hyperSpec objects.

**Usage**

```r
## S4 method for signature 'hyperSpec,hyperSpec'
Compare(e1, e2)

## S4 method for signature 'hyperSpec,numeric'
Compare(e1, e2)

## S4 method for signature 'hyperSpec,matrix'
Compare(e1, e2)

## S4 method for signature 'numeric,hyperSpec'
Compare(e1, e2)

## S4 method for signature 'matrix,hyperSpec'
Compare(e1, e2)

## S4 method for signature 'hyperSpec,hyperSpec'
all.equal(target, current, ..., 
  check.attributes = FALSE, check.names = FALSE, 
  check.column.order = FALSE, check.label = FALSE, 
  tolerance = hy.getOption("tolerance"), 
  wl.tolerance = hy.getOption("wl.tolerance"))
```

**Arguments**

- `e1, e2` Either two hyperSpec objects or one hyperSpec object and matrix of same size as `hyperSpec[[[]]]` or a scalar (numeric of length 1). As hyperSpec objects must have numeric spectra matrices, the resulting matrix of the comparison is returned directly.
- `target, current` two hyperSpec objects that are tested for equality
- `...` handed to `all.equal` when testing the slots of the hyperSpec objects
check.attributes, check.names
  see all.equal

check.column.order
  If two objects have the same data, but the order of the columns (determined by the names) differs, should they be regarded as different?

check.label
  Should the slot label be checked?
  If the labels differ only in the order of their entries, they are considered equal.

tolerance, wl.tolerance
  tolerances for checking wavelengths and data, respectively

Details

all.equal checks the equality of two hyperSpec objects.

The comparison operators >, <, >=, <=, and != work on the spectra matrix of the hyperSpec object. They have their usual meaning (see Comparison). The operators work also with one hyperSpec object and a numeric (scalar) object or a matrices of the same size as the spectra matrix of the hyperSpec object.

With numeric vectors sweep might be more appropriate.

If you want to calculate on the data.frame hyperSpec@data, you have to do this directly on hyperSpec@data.

Value

a logical matrix for the comparison operators.

all.equal returns either TRUE, or a character vector describing the differences. In conditions, the result must therefore be tested with isTRUE.

Author(s)

C. Beleites

See Also

sweep-methods for calculations involving a vector and the spectral matrix.
S4groupGeneric for group generic methods.
Comparison for the base comparison functions.
Arith for arithmetic operators, Math for mathematical group generic functions (groups Math and Math2) working on hyperSpec objects.
all.equal and isTRUE

Examples

flu [,445 ~ 450] > 300

all (flu == flu[[]])
cov, hyperSpec, missing-method

Covariance matrices for hyperSpec objects

Description

Covariance matrices for hyperSpec objects

Usage

```r
## S4 method for signature 'hyperSpec,missing'
cov(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))
```

```r
pooled.cov(x, groups, ..., regularize = 1e-05 * max(abs(COV)))
```

Arguments

- `x`: hyperSpec object
- `y`: not supported
- `use, method`: handed to `cov`
- `groups`: factor indicating the groups
- `...`: ignored
- `regularize`: regularization of the covariance matrix. Set 0 to switch off
  pooled.cov calculates pooled covariance like e.g. in LDA.

Value

covariance matrix of size `nwl(x) x nwl(x)`

Author(s)

C. Beleites

See Also

cov

Examples

```r
image (cov (chondro))
pcov <- pooled.cov (chondro, chondro$clusters)
plot (pcov$means)
image (pcov$COV)
```
decomposition

**Convert Principal Component Decomposition or the like into a hyperspec object**

**Description**

Decomposition of the spectra matrix is a common procedure in chemometric data analysis. Scores and loadings convert the result matrices into new hyperspec objects.

**Usage**

```r
decomposition(object, x, wavelength = seq_len(ncol(x)), label.wavelength, label.spc, scores = TRUE, retain.columns = FALSE, ...)
```

**Arguments**

- `object` A hyperspec object.
- `x` matrix with the new content for `object$data$spc`. Its size must correspond to rows (for scores) and to either columns or rows (for loadings) of `object`.
- `wavelength` for a scores-like `x`: the new `object$wavelength`.
- `label.wavelength` The new label for the wavelength axis (if `x` is scores-like). If not given, the label of `object` is kept.
- `label.spc` The new label for the spectra matrix. If not given, the label of `object` is kept.
- `scores` is `x` a scores-like matrix?
- `retain.columns` for loading-like decomposition (i.e. `x` holds loadings, pure component spectra or the like), the data columns need special attention. Columns with different values across the rows will be set to `NA` if `retain.columns` is `TRUE`, otherwise they will be deleted.
- `...` ignored.

**Details**

Multivariate data are frequently decomposed by methods like principal component analysis, partial least squares, linear discriminant analysis, and the like. These methods yield latent spectra (or latent variables, loadings, components, ...) that are linear combination coefficients along the wavelength axis and scores for each spectrum and loading.

The loadings matrix gives a coordinate transformation, and the scores are values in that new coordinate system.

The obtained latent variables are spectra-like objects: a latent variable has a coefficient for each wavelength. If such a matrix (with the same number of columns as `object` has wavelengths) is given to `decomposition` (also setting scores = `FALSE`), the spectra matrix is replaced by `x`. Moreover, all columns of `object$data` that did not contain the same value for all spectra are set
to NA. Thus, for the resulting hyperSpec object, plotspec and related functions are meaningful. plotmap cannot be applied as the loadings are not laterally resolved.

The scores matrix needs to have the same number of rows as object has spectra. If such a matrix is given, decomposition will replace the spectra matrix is replaced by x and object@wavelength by wavelength. The information related to each of the spectra is retained. For such a hyperSpec object, plotmap and plotc and the like can be applied. It is also possible to use the spectra plotting, but the interpretation is not that of the spectrum any longer.

Value

A hyperSpec object, updated according to x

Author(s)

C. Beleites

See Also

See %*% for matrix multiplication of hyperSpec objects.
See e.g. prcomp and princomp for principal component analysis, and package pls for Partial Least Squares Regression.

Examples

pca <- prcomp (flu)
pca.loadings <- decomposition (flu, t (pca$rotation), scores = FALSE)
pca.center <- decomposition (flu, pca$center, scores = FALSE)
pca.scores <- decomposition (flu, pca$x)

plot (pca.center)
plot (pca.loadings, col = c ("red", "gray50"))
plotc (pca.scores, groups = .wavelength)
Usage

```r
## S4 method for signature 'hyperSpec'
dimnames(x)

## S4 method for signature 'hyperSpec'
rownames(x, do.NULL = TRUE, prefix = "row")

## S4 replacement method for signature 'hyperSpec'
rownames(x) <- value

## S4 method for signature 'hyperSpec'
colnames(x, do.NULL = TRUE, prefix = "col")

## S4 replacement method for signature 'hyperSpec'
colnames(x) <- value
```

Arguments

- `x` the hyperSpec object
- `do.NULL` handed to `rownames` or `colnames`: logical. Should this create names if they are NULL?
- `prefix` handed to `rownames` or `colnames`
- `value` the new names

Author(s)

C. Beleites

See Also

- `wl` for the wavelength dimension
- `dimnames`
- `rownames`
- `colnames`

Examples

```r
dimnames (flu)
rownames (flu)
colnames (chondro)
```
empty

Empty hyperSpec object

Description
Empty produces an hyperSpec object with the same columns and wavelengths as x. The new object will either contain no rows at all (default), or the given number of rows with all data initialized to spc and extra, respectively.

Usage
`empty(x, nrow = 0, spc = NA, extra = NA)`

Arguments
- **x**: hyperSpec object
- **nrow**: number of rows the new object should have
- **spc**: value to initialize the new spectra matrix with
- **extra**: value to initialize the new extra data with

Author(s)
C. Beleites

Examples
`empty (chondro, nrow = 2, spc = 0)`

flu

Quinine Fluorescence Spectra Fluorescence spectra of different dilutions of quinine forming a calibration set.

Description
See the vignette: vignette("flu", package = "hyperSpec")

Format
The data set has 6 fluorescence emission spectra measured on quinine concentrations between 0.05 mg/l and 0.30 mg/l. Each spectrum consists of 181 data points in the range of 405 nm to 495 nm.

Author(s)
M. Kammer and C. Beleites
Examples

flu
plot (flu)
plotc (flu)

hy.getOptions
Options for package hyperSpec Functions to access and set hyperSpec’s options.

Description
Currently, the following options are defined:

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value (range)</th>
<th>Description</th>
<th>Used by</th>
</tr>
</thead>
<tbody>
<tr>
<td>debuglevel</td>
<td>0 (1L 2L)</td>
<td>amount of debugging information produced</td>
<td>spc.identify, various file import functions</td>
</tr>
<tr>
<td>gc</td>
<td>FALSE</td>
<td>triggers frequent calling of gc ()</td>
<td>read.ENVI, normalize, various file import functions</td>
</tr>
<tr>
<td>file.remove.emptyspc</td>
<td>TRUE</td>
<td>remove empty spectra directly on file import</td>
<td>normalize, various file import functions</td>
</tr>
<tr>
<td>file.keep.name</td>
<td>TRUE</td>
<td>always create filename column</td>
<td>all.equal, collapse</td>
</tr>
<tr>
<td>tolerance</td>
<td>sqrt (.Machine$double.eps)</td>
<td>tolerance for numerical comparisons</td>
<td></td>
</tr>
<tr>
<td>wl.tolerance</td>
<td>sqrt (.Machine$double.eps)</td>
<td>tolerance for comparisons of the wavelength axis</td>
<td></td>
</tr>
</tbody>
</table>

Usage

hy.getOptions(...)  
hy.getOption(name)  
hy.setOptions(...)  

Arguments

...  
hy.setOptions: pairs of argument names and values.  
hy.getOptions: indices (or names) of the options.  
name the name of the option

Details

hy.setOptions will discard any values that were given without a name.

Value
hy.unittest

- `hy.getOptions` returns a list of all options
- `hy.setOptions` invisibly returns a list with the options
- `hyTumblr` returns the value of the requested option

**Author(s)**

C. Beleites

**Examples**

```r
hy.getOptions()
```

**Description**

hyperSpec unit tests If `svUnit` is available, run the unit tests and display the results.

**Usage**

```
hy.unittest()
```

**Value**

NA if `svUnit` is not available, otherwise TRUE if all tests are passed successfully. If a test fails, `hy.unittest` stops with an error.

**Author(s)**

C. Beleites

**See Also**

- `svUnit`

**Examples**

```r
if (require (svUnit)){
  hy.unittest()
}
```
hyperSpec-class

Class "hyperSpec" This class handles hyperspectral data sets, i.e. spatially or time-resolved spectra, or spectra with any other kind of information associated with the spectra.

Description

The spectra can be data as obtained in XRF, UV/VIS, Fluorescence, AES, NIR, IR, Raman, NMR, MS, etc.

Details

More generally, any data that is recorded over a discretized variable, e.g. absorbance = f (wavelength), stored as a vector of absorbance values for discrete wavelengths is suitable.

Slots

- `wavelength`: wavelengths (wavenumbers, frequencies, etc.) for each of the columns of the spectra matrix
- `data`: the data (extra data and spectra matrix)
- `label`: expressions for column labels (incl. units). The label of the wavelength axis is in the special element `.wavelength`.
- `log`: deprecated.

Note

Please note that the logbook is now removed.

Author(s)

C. Beleites

See Also

See the vignette "introduction" for an introduction to hyperSpec from a spectroscopic point of view.

Examples

```r
showClass("hyperSpec")
## Not run: vignette("introduction")
```
initialize

Creating a hyperSpec Object Like other S4 objects, a hyperSpec object can be created by new. The hyperSpec object is then initialized using the given parameters.

Description

If option gc is TRUE, the initialization will have frequent calls to gc () which can help to avoid swapping or running out of memory.

Usage

```r
## S4 method for signature 'hyperSpec'
initialize(.Object, spc = NULL, data = NULL,
           wavelength = NULL, labels = NULL)
```

Arguments

- `.Object` the new hyperSpec object.
- `spc` the spectra matrix.
  - spc does not need to be a matrix, it is converted explicitly by I (as.matrix (spc)).
- `data` data.frame, possibly with the spectra in data$spc, and further variates in more columns. A matrix can be entered as one column of a data frame by: data.frame (spc = I (as.matrix (spc))). However, it will usually be more convenient if the spectra are given in spc
- `wavelength` The wavelengths corresponding to the columns of data. If no wavelengths are given, an appropriate vector is derived from the column names of data$spc. If this is not possible, 1 : ncol (data$spc) is used instead.
- `labels` A list containing the labels for the columns of the data slot of the hyperSpec object and for the wavelength (in label$. wavelength). The labels should be given in a form ready for the text-drawing functions (see plotmath).
  - If label is not given, a list containing NULL for each of the columns of data and wavelength is used.

Author(s)

C.Beleites

See Also

- `new` for more information on creating and initializing S4 objects.
- `plotmath` on expressions for math annotations as for slot label.
- `hy.setOptions`
Examples

```r
new("hyperSpec")

spc <- matrix(rnorm(12), ncol = 4)
new("hyperSpec", spc = spc)
new("hyperSpec", data = data.frame(x = letters[1:3]),
   spc = spc)

colnames(spc) <- 600:603
new("hyperSpec", spc = spc)  # wavelength taken from colnames (spc)

# given wavelengths precede over colnames of spc
new("hyperSpec", spc = spc, wavelength = 700:703)

# specifying labels
h <- new("hyperSpec", spc = spc, data = data.frame(pos = 1:3),
   label = list(spc = "I / a.u.",
               wavelength = expression(tilde(nu) / cm^-1),
               pos = expression("/" (x, mu=nm)))
)

plot(h)
plotc(h, spc ~ pos)
```

---

labels<-  
*Get and Set Labels of a hyperSpec Object*

value may be a list or vector of labels giving the new label for each of the entries specified by which.

---

Description

The names of the labels are the same as the colnames of the data.frame. The label for the wavelength axis has the name .wavelength.

Usage

```r
labels(object, which = NULL, ...) <- value
```

## S4 method for signature 'hyperSpec'

```r
labels(object, which = bquote(), drop = TRUE, ...,
   use.colnames = TRUE)
```

Arguments

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>a hyperSpec object</td>
</tr>
<tr>
<td>which</td>
<td>numeric or character to specify the label(s)</td>
</tr>
<tr>
<td>...</td>
<td>ignored</td>
</tr>
</tbody>
</table>
value the new label(s)
drop if the result would be a list with only one element, should the element be returned instead?
use.colnames should missing labels be replaced by column names of the extra data?

Details
The labels should be given in a form ready for the text-drawing functions (see plotmath), e.g. as expression or a character.

Value
labels<- returns a hyperSpec object.
labels returns a list of labels. If drop is TRUE and the list contains only one element, the element is returned instead.

Author(s)
C. Beleites

See Also
labels

Examples
labels (flu, "c") <- expression ("/" ("c", "mg / l"))

labels (chondro)

laser Laser Emission A time series of an unstable laser emission.

Description
see the Vignette

Format
The data set consists of 84 laser emission spectra measured during 95 min. Each spectrum has 36 data points in the range of 404.5 nm to 405.8 nm.

Author(s)
C. Beleites
Examples

laser

cols <- c("black", "blue", "darkgreen", "red")
wavelength <- c(405.0, 405.1, 405.3, 405.4)
plotspc(laser, axis.args=list(x=list(at=seq(404.5, 405.8, .1))))
for (i in seq_along(wavelength))
  abline(v=wavelength[i], col=cols[i], lwd=2, lty=2)

plotc(laser[., wavelength], spc~t, groups=.wavelength, type="b",
col=cols)

## Not run: vignette("laser", package="hyperSpec")

---

legendright

Plot multivariate data into colour channels

Description

plot graph with legend right of it
plot multivariate data into colour channels using geom_tile
normalize.colrange normalizes the range of each column to [0, 1]
normalize.range normalizes the range of all columns to [0, 1]
normalize.null does not touch the values
normalize.minmax normalizes the range of each column j to [min_j, max_j]
legends for mixed colour plots

Usage

legendright(p, l, legend.width = 8, legend.unit = "lines")

qmixtile(object, purecol = stop("pure component colors needed."),
  mapping = aes_string(x = "x", y = "y", fill = "spc"), ..., 
  map.tileonly = FALSE)

normalize.colrange(x, na.rm = TRUE, legend = FALSE, n = 100, ...)
normalize.range(x, na.rm = TRUE, legend = FALSE, n = 100, ...)
normalize.null(x, na.rm = TRUE, legend = FALSE, n = 100, ...)
normalize.minmax(x, min = 0, max = 1, legend = FALSE, n = 100, ...)
qmixlegend(x, purecol, dx = 0.33, ny = 100, labels = names(purecol),
normalize = normalize.colrange, ...)

```
colmix.rgb(x, purecol, against = 1, sub = TRUE,
            normalize = normalize.colrange, ...)
```

### Arguments
- **p**: plot object
- **l**: legend object
- **legend.width, legend.unit**: size of legend part
- **object**: matrix to be plotted with mixed colour channels
- **purecol**: pure component colours, names determine legend labels
- **mapping**: see `geom_tile`
- **...**: `qmixtile`: handed to `colmix.rgb`
- **map.tileonly**: if TRUE, mapping will be handed to `geom_tile` instead of `ggplot`
- **x**: matrix with component intensities in columns
- **na.rm**: see `link[base](min)`
- **legend**: should a legend be produced instead of normalized values?
- **n**: of colours to produce in legend
- **min**: numeric with value corresponding to "lowest" colour for each column
- **max**: numeric with value corresponding to "highest" colour for each column
- **dx**: width of label bar
- **ny**: number of colours in legend
- **labels**: component names
- **normalize**: function to normalize the values.
- **against**: value to mix against (for sub = TRUE only, 1 = white, 0 = black)
- **sub**: subtractive color mixing?

### Value
- invisible NULL
- list with components ymin, max and fill to specify value and fill colour value (still numeric!) for the legend, otherwise the normalized values
- `ggplot` object with legend
- character with colours

### Author(s)
Claudia Beleites
Claudia Beleites
Claudia Beleites
Description

find an evenly spaced grid for x

Usage

```
makeraster(x, startx, d, newlevels, tol = 0.1)
fitraster(x, tol = 0.1)
```

Arguments

- **x** numeric to be fitted with a raster
- **startx** starting point ("origin") for calculation of the raster
- **d** step size of the raster
- **newlevels** levels of the raster
- **tol** tolerance for rounding to new levels: elements of x within tol of the distance between the levels of the new grid are rounded to the new grid point.

Details

- **makeraster** fits the data to the specified raster.
- **fitraster** tries different raster parameter and returns the raster that covers most of the x values:
  The differences between the values of x are calculated (possible step sizes). For each of those step sizes, different points are tried (until all points have been covered by a raster) and the parameter combination leading to the best coverage (i.e. most points on the grid) is used.
  Note that only differences between the sorted values of x are considered as step size.

Value

- list with elements
  - x the values of x, possibly rounded to the raster values
  - levels the values of the raster

Author(s)

- Claudia Beleites
Examples

```r
x <- c(sample(1:20, 10), (0 : 5) + 0.5)
raster <- makeraster (x, x [1], 2)
raster
plot (x)
abline (h = raster$levels, col = "#00000040")

## unoccupied levels
missing <- setdiff(raster$levels, raster$x)
abline (h = missing, col = "red")

## points actually on the raster
onraster <- raster$x %in% raster$levels
points (which (onraster), raster$x [onraster], col = "blue", pch = 20)
```

```r
raster <- fitraster (x)
raster
plot (x)
abline (h = raster$levels, col = "#00000040")

## unoccupied levels
missing <- setdiff(raster$levels, raster$x)
abline (h = missing, col = "red")

## points actually on the raster
onraster <- raster$x %in% raster$levels
points (which (onraster), raster$x [onraster], col = "blue", pch = 20)
```

```r
x <- c(sample(1:20, 10), (0 : 5) + 0.45)
raster <- fitraster (x)
raster
plot (x)
abline (h = raster$levels, col = "#00000040")

## unoccupied levels
missing <- setdiff(raster$levels, raster$x)
abline (h = missing, col = "red")

## points actually on the raster
onraster <- raster$x %in% raster$levels
points (which (onraster), raster$x [onraster], col = "blue", pch = 20)
```

---

**map.sel.poly**

*Interactively select a polygon (grid graphics) and highlight points*

**Description**

Click the points that should be connected as polygon. Input ends with right click (see `grid.locator`). Polygon will be drawn closed.
Usage

```r
map.sel.poly(data, pch = 19, size = 0.3, ...) sel.poly(pch = 19, size = 0.3, ...)
```

Arguments

- `data`: hyperSpec object for plotting map or list returned by `plotmap`
- `pch`: symbol to display the points of the polygon for `sel.poly`
- `size`: size for polygon point symbol for `sel.poly`
- `...`: further arguments for `grid.points` and `grid.lines`

Details

`map.sel.poly` is a convenience wrapper for `plotmap`, `sel.poly`, and `point.in.polygon`. For customized plotting, the plot can be produced by `plotmap`, `plotvoronoi` or `levelplot`, and the result of that plot command handed over to `map.sel.poly`, see the example below.

If even more customized plotting is required, `sel.poly` should be used (see example).

Value

- `map.sel.poly`: array of indices for points within the selected polygon
- `sel.poly`: n x 2 matrix with the corner points of the polygon

Author(s)

Claudia Beleites, Sebastian Mellor
Claudia Beleites

See Also

- `grid.locator`, `map.identify`
- `grid.locator`

Examples

```r
if (interactive ()){
  ## convenience wrapper
  map.sel.poly (chondro)

  ## customized version
  data <- sample (chondro [, , 1004 - 2i - 1004 + 2i], 300)

  plotdata <- plotvoronoi (data, clusters ~ y * x, col.regions = alois.palette ())
  print (plotdata)
  map.sel.poly (plotdata)

  ## even more customization:
```
mark.dendrogram

mark.dendrogram (data)

## interactively retrieve polygon
polygon <- sel.poly()

## find data points within polygon
require("sp")
i.sel <- which(point.in.polygon(data$x, data$y, polygon [,1], polygon [,2]) > 0)

## work with selected points
grid.points (unit(data$x [i.sel], "native"), unit(data$y [i.sel], "native"))

mark.dendrogram

### Mark groups in hclust dendrograms

**Description**

Groups are marked by colored rectangles as well as by their levels.

**Usage**

```r
mark.dendrogram(dendrogram, groups, col = seq_along(unique(groups)),
pos.marker = 0, height = 0.025 * max(dendrogram$height), pos.text = -2.5
* height, border = NA, text.col = "black", label, label.right = TRUE,
...)```

**Arguments**

- `dendrogram`: the dendrogram
- `groups`: factor giving the the groups to mark
- `col`: vector with colors for each group
- `pos.marker`: top of the marker rectangle
- `height`: height of the marker rectangle
- `pos.text`: position of the text label
- `border`: see `text`
- `text.col`: color (vector) giving the color for the text markers
- `label`: side label see example
- `label.right`: should the side labels be at the right side?
- `...`: handed to `rect` and `text`

**Details**

The dendrogram should be plotted separately, see the example.
markpeak

Mark peak Marks location of the first spectrum at the data point closest to the specified position on the current plot.

Description

Mark peak

Marks location of the first spectrum at the data point closest to the specified position on the current plot.

Usage

markpeak(spc, xpos, col = "red")

Arguments

spc the hyperSpec object
xpos position of the peak(s) in current x-axis units
col color of the markers and text

Author(s)

R. Kiselev
Math2.hyperSpec-method

Math Functions for hyperSpec Objects

Description

Mathematical functions for hyperSpec Objects.

Usage

```r
## S4 method for signature 'hyperSpec'
Math2(x, digits)
```

```r
## S4 method for signature 'hyperSpec'
log(x, base = exp(1), ...)
```

```r
## S4 method for signature 'hyperSpec'
Math(x)
```

Arguments

- `x`: the hyperSpec object
- `digits`: integer stating the rounding precision
- `base`: base of logarithm
- `...`: ignored

Details

The functions abs, sign, sqrt, floor, ceiling, trunc, round, signif, exp, log, expm1, log1p, cos, sin, tan, acos, asin, atan, cosh, sinh, tanh, acosh, asinh, atanh, lgamma, gamma, digamma, trigamma, cumsum, cumprod, cummax, cummin for hyperSpec objects.

Value

a hyperSpec object

Author(s)

C. Beleites
See Also

S4groupGeneric for group generic methods.
Math for the base math functions.
Arith for arithmetic operators, Comparison for comparison operators, and Summary for group generic functions working on hyperSpec objects.

Examples

log (flu)

---

**matlab.palette**

Matlab-like Palettes Two palettes going from blue over green to red, approximately as the standard palette of Matlab does. The second one has darker green values and is better suited for plotting lines on white background.

Description

Matlab-like Palettes Two palettes going from blue over green to red, approximately as the standard palette of Matlab does. The second one has darker green values and is better suited for plotting lines on white background.

Usage

matlab.palette(n = 100, ...)
matlab.dark.palette(n = 100, ...)
alois.palette(n = 100, ...)

Arguments

n the number of colors to be in the palette.
... further arguments are handed to rainbow(alois.palette: colorRampPalette)

Value

A vector containing the color values in the form "\#rrbbgaaa".

Author(s)

C. Beleites and A. Bonifacio
See Also

rainbow

Examples

plotmap (chondro [, , 778], col.regions = matlab.palette ())

plot (flu, col = matlab.dark.palette (ncol (flu)))
plotmap (chondro, col = alois.palette)

mean_sd

Mean and Standard Deviation Calculate mean and standard deviation, and mean, mean \pm \text{one standard deviation}, respectively.

Description

These functions are provided for convenience.

Usage

```r
## S4 method for signature 'numeric'
mean_sd(x, na.rm = TRUE, ...)

## S4 method for signature 'matrix'
mean_sd(x, na.rm = TRUE, ...)

## S4 method for signature 'hyperSpec'
mean_sd(x, na.rm = TRUE, ...)

## S4 method for signature 'numeric'
mean_pm_sd(x, na.rm = TRUE, ...)

## S4 method for signature 'matrix'
mean_pm_sd(x, na.rm = TRUE, ...)

## S4 method for signature 'hyperSpec'
mean_pm_sd(x, na.rm = TRUE, ...)

## S4 method for signature 'hyperSpec'
quantile(x, probs = seq(0, 1, 0.5), na.rm = TRUE,
           names = "num", ...)
```
Arguments

- `x`: a numeric vector
- `na.rm`: handed to `mean` and `sd`
- `...`: ignored (needed to make function generic)
- `probs`: the quantiles, see `quantile`
- `names`: "pretty" results in percentages (like `quantile`'s `names = TRUE`), "num" results in the row names being `as.character` (`probs`) (good for ggplot2 getting the order of the quantiles right). Otherwise, no names are assigned.

Value

- `mean_sd` returns a vector with two values (mean and standard deviation) of `x`.
- `mean_sd` (matrix) returns a matrix with the mean spectrum in the first row and the standard deviation in the 2nd.
- `mean_sd` returns a hyperSpec object with the mean spectrum in the first row and the standard deviation in the 2nd.
- `mean_pm_sd` returns a vector with 3 values: mean - 1 sd, mean, mean + 1 sd
- `mean_pm_sd` (matrix) returns a matrix containing mean - sd, mean, and mean + sd rows.

For hyperSpec objects, `mean_pm_sd` returns a hyperSpec object containing mean - sd, mean, and mean + sd spectra.

For hyperSpec object, `mean` returns a hyperSpec object containing the mean spectrum.

For hyperSpec object, `quantile` returns a hyperSpec object containing the respective quantile spectra.

Author(s)

C. Beleites

See Also

- `mean`, `sd`
- `mean`, `sd`
- `quantile`

Examples

- `mean_sd (flu [, , 405 - 410])`
- `mean_sd (flu$spc)`
- `mean_sd (flu)`
- `mean_pm_sd (flu$c)`
Merges two hyperSpec objects and `cbinds` their spectra matrices.

```r
## S4 method for signature 'hyperSpec,hyperSpec'
merge(x, y, ...)  
```

### Arguments

- `x`  
  a hyperSpec object

- `y`  
  a hyperSpec object

- `...`  
  handed to `merge.data.frame`

### Details

After merging, the spectra matrix can contain duplicates, and is not ordered according to the wavelength.

If the wavelength axis should be ordered, use `orderwl`.

### Author(s)

C. Beleites

### See Also

- `merge`
- `collapse` combines hyperSpec objects that do not share the wavelength axis. `rbind`, and `cbind` for combining hyperSpec objects that.
Examples

merge (chondro [1:10, 600], chondro [5:15, 600], by = c("x", "y"))$

tmp <- merge (chondro [1:10, 610], chondro [5:15, 610],
             by = c("x", "y"), all = TRUE)

tmp$

wl (tmp)

## remove duplicated wavelengths:
approxfun <- function (y, wl, new.wl){
  approx (wl, y, new.wl, method = "constant",
          ties = function (x) mean (x, na.rm = TRUE)
        )$y
}

merged <- merge (chondro [1:7, 610 - 620], chondro [5:10, 615 - 625], all = TRUE)
merged$

merged <- apply (merged, 1, approxfun,
                 wl = wl (merged), new.wl = unique (wl (merged)),
                 new.wavelength = "new.wl")
merged$

ncol,hyperSpec-method
The Number of Rows (Spectra), Columns, and Data Points per Spectrum of an hyperSpec Object

Description

ncol returns the number of columns in x@data. I.e. the number of columns with additional information to each spectrum (e.g. "x", "y",...,) + 1 (for column spc containing the spectra).
nrow yields the number of rows in x@data, i.e. the number of spectra in the hyperSpec object.
nwl returns the number of columns in x@data$spc, i.e. the length of each spectrum.
dim returns all three values in a vector.
length is a synonym for nrow. It is supplied so that seq_along (x) returns a sequence to index each spectrum.

Usage

## S4 method for signature 'hyperSpec'
ncol(x)

## S4 method for signature 'hyperSpec'
nrow(x)
nwl(x)
### S4 method for signature 'hyperSpec'

`dim(x)`

### S4 method for signature 'hyperSpec'

`length(x)`

#### Arguments

- `x`: a `hyperSpec` object

#### Value

- `nrow`, `ncol`, `nwl`, and `length`, return an integer.
- `dim` returns a vector of length 3.

#### Author(s)

C. Beleites

#### See Also

- `ncol`
- `nrow`
- `dim`
- `length`

#### Examples

```r
ncol(chondro)
nrow(chondro)

nwl(chondro)
dim(chondro)
length(chondro)
```
Usage

```r
normalize01(x, ...)
```

```r
## S4 method for signature 'matrix'
normalize01(x, tolerance = hy.getOption("tolerance"))
```

```r
## S4 method for signature 'numeric'
normalize01(x, tolerance = hy.getOption("tolerance"))
```

```r
## S4 method for signature 'hyperSpec'
normalize01(x, ...)
```

Arguments

- `x` vector with values to transform
- `...` additional parameters such as `tolerance` handed down.
- `tolerance` tolerance level for determining what is 0 and 1

Details

The input `x` is mapped to [0, 1] by subtracting the minimum and subsequently dividing by the maximum. If all elements of `x` are equal, 1 is returned.

Value

- vector with `x` values mapped to the interval [0, 1]

Author(s)

C. Beleites

See Also

- `wl.eval`, `vanderMonde`

Description

```
Sorting the Wavelengths of a hyperSpec Object Rearranges the hyperSpec object so that the wavelength vector is in increasing (or decreasing) order.
```

Usage

```r
orderwl(x, na.last = TRUE, decreasing = FALSE)
```
Arguments

x The hyperSpec object.

na.last, decreasing

Handed to order.

Value

A hyperSpec object.

Author(s)

C. Beleites

See Also

order

Examples

## Example 1: different drawing order in plotspc
spc <- new("hyperSpec", spc = matrix(rnorm (5) + 1:5, ncol = 5))
spc <- cbind(spc, spc+.5)

plot (spc, "spc")
text (wl (spc), spc [[]], as.character (1:10))
spc <- orderwl (spc)
plot (spc, "spc")
text (wl (spc), spc [[]], as.character (1:10))

## Example 2
spc <- new("hyperSpec", spc = matrix(rnorm (5)*2 + 1:5, ncol = 5))
spc <- cbind (spc, spc)

plot (seq_len(nwl(spc)), spc[[]], type = "b")
spc[[]]

spc <- orderwl (spc)
lines (seq_len(nwl(spc)), spc[[]], type = "l", col = "red")
spc[[]]

paracetamol  Paracetamol Spectrum A Raman spectrum of a paracetamol tablet.

Description

Paracetamol Spectrum A Raman spectrum of a paracetamol tablet.
The spectrum was acquired with a Renishaw InVia spectrometer from 100 to 3200 cm^{-1} in step scan mode. Thus the spectrum has several overlapping wavelength regions.

Author(s)

C. Beleites

Examples

```
paracetamol

plot (paracetamol)
plotspc (paracetamol, c (min ~ 1750, 2800 ~ max), xoffset = 800,
wl.reverse = TRUE)
```

---

**pearson.dist**

*Distance based on Pearson’s $R^2$*

**Description**

The calculated distance is $D^2 = \frac{1-COR(x^t)}{2}$

**Usage**

```
pearson.dist(x)
```

**Arguments**

- **x**: a matrix

**Details**

The distance between the rows of x is calculated. The possible values range from 0 (perfectly correlated) over 0.5 (uncorrelated) to 1 (perfectly anti-correlated).

**Value**

- distance matrix (distance object)

**Author(s)**

C. Beleites

**References**

S. Theodoridis and K. Koutroumbas: Pattern Recognition, 3rd ed., p. 495
plot-methods

See Also

as.dist

Examples

pearson.dist (flu [[[]])
pearson.dist (flu)

plot-methods  Plotting hyperSpec Objects

Description

Plotting hyperSpec objects. The plot method for hyperSpec objects is a switchyard to plotspec, plotmap, and plotc.

Usage

```r
## S4 method for signature 'hyperSpec,missing'
plot(x, y, ...)

## S4 method for signature 'hyperSpec,character'
plot(x, y, ...)
```

Arguments

- `x` the hyperSpec object
- `y` selects what plot should be produced
- `...` arguments passed to the respective plot function

Details

It also supplies some convenient abbreviations for much used plots.

If `y` is missing, plot behaves like `plot (x, y = "spc")`. Supported values for `y` are:

"spc" calls plotspec to produce a spectra plot.
"spcmeansd" plots mean spectrum +/- one standard deviation
"spcprctile" plots 16th, 50th, and 84th percentile spectra. If the distributions of the intensities at all wavelengths were normal, this would correspond to "spcmeansd". However, this is frequently not the case. Then "spcprctile" gives a better impression of the spectral data set.
"spcprctile5" like "spcprctile", but additionally the 5th and 95th percentile spectra are plotted.
"map" calls plotmap to produce a map plot.
"voronoi" calls `plotvoronoi` to produce a Voronoi plot (tesselated plot, like "map" for hyperSpec objects with uneven/non-rectangular grid).

"mat" calls `plotmat` to produce a plot of the spectra matrix (not to be confused with `matplot`).

"c" calls `plotc` to produce a calibration (or time series, depth-profile, or the like)

"ts" plots a time series: abbreviation for `plotc` (x, use.c = "t")

"depth" plots a depth profile: abbreviation for `plotc` (x, use.c = "z")

Author(s)

C. Beleites

See Also

`plotspc` for spectra plots (intensity over wavelength),
`plotmap` for plotting maps, i.e. color coded summary value on two (usually spatial) dimensions.

`plotc`

`plot`

Examples

```r
plot(flu)
plot(flu, "c")
plot(laser, "ts")
```

```r
spc <- apply(chondro, 2, quantile, probs = 0.05)
spc <- sweep(chondro, 2, spc, "+-")
plot(spc, "spcprctl5")
plot(spc, "spcprctile")
plot(spc, "spcmeansd")
```

---

**plotc**

*Calibration- and Timeseries Plots, Depth-Profiles and the like*

`plotc` plots intensities of a hyperSpec object over another dimension such as concentration, time, or a spatial coordinate.

---

**Description**

If `func` is not `NULL`, the summary characteristic is calculated first by applying `func` with the respective arguments (in `func.args`) to each of the spectra. If `func` returns more than one value (for each spectrum), the different values end up as different wavelengths.
plotc

Usage

plotc(object, model = spc ~ c, groups = NULL, func = NULL,
       func.args = list(), ...)

Arguments

object       the hyperSpec object
model        the lattice model specifying the plot
groups       grouping variable, e.g. .wavelength if intensities of more than one wavelength
             should be plotted
func         function to compute a summary value from the spectra to be plotted instead of
             single intensities
func.args    further arguments to func
...          further arguments to xyplot.

Details

If the wavelength is not used in the model specification nor in groups, nor for specifying subsets,
and neither is func given, then only the first wavelength’s intensities are plotted and a warning is
issued.

The special column names .rownames and .wavelength may be used.

The actual plotting is done by xyplot.

Author(s)

C. Beleites

See Also

xyplot

Examples

```r
## example 1: calibration of fluorescence
plotc (flu) ## gives a warning
plotc (flu, func = mean)
plotc (flu, func = range, groups = .wavelength)

plotc (flu[,450], ylab = expression (I ["450 nm"] / a.u.))

# calibration <- lm (spc ~ c, data = flu[,450]$.)
summary (calibration)
plotc (flu [, 450], type = c("p", "r"))
```

plotmap

Plot a Map and Identify/Select Spectra in the Map levelplot functions for hyperSpec objects. An image or map of a summary value of each spectrum is plotted. Spectra may be identified by mouse click.

Description

The model can contain the special column name .wavelength to specify the wavelength axis.

Usage

plotmap(object, model = spc ~ x * y, func = mean, func.args = list(), ...)

## S4 method for signature 'hyperSpec,missing'
levelplot(x, data, ...)

## S4 method for signature 'formula,hyperSpec'
levelplot(x, data, transform.factor = TRUE, ...,
    contour = FALSE, useRaster = !contour)

map.identify(object, model = spc ~ x * y, voronoi = FALSE, ...,
    tol = 0.02, warn = TRUE)
plotmap

plotvoronoi(object, model = spc ~ x * y, use.tripack = FALSE, mix = FALSE, ...

Arguments

object, data the hyperSpec object
model, x formula specifying the columns of object that are to be displayed by levelplot
func, func.args

Before plotting, plotmap applies function func with the arguments given in the list func.args to each of the spectra. Thus a single summary value is displayed for each of the spectra.

This can be suppressed manually by setting func to NULL. It is automatically suppressed if .wavelength appears in the formula.

... further arguments are passed down the call chain, and finally to levelplot
transform.factor

If the color-coded variable is a factor, should trellis.factor.key be used to compute the color coding and legend?
contour, useRaster see levelplot
voronoi Should the plot for identifying spectra by mouse click be produced by plotmap (default) or plotvoronoi?
tol tolerance for map.identify as fraction of the viewport (i.e. in "npc" units)
warn should a warning be issued if no point is within the specified tolerance? See also details.
use.tripack Whether package tripack should be used for calculating the voronoi polygons. If FALSE, package deldir is used instead. See details.
mix For Voronoi plots using package tripack, I experienced errors if the data was spatially ordered. Randomly rearranging the rows of the hyperSpec object circumvents this problem.

Details

plotmap, map.identify, and the levelplot methods internally use the same gateway function to levelplot. Thus transform.factor can be used with all of them and the panel function defaults to panel.levelplot.raster for all three. Two special column names, .rownames and .wavelength may be used.

levelplot plots the spectra matrix.

plotvoronoi calls plotmap with different default settings, namely the panel function defaults to panel.voronoi. panel.voronoi depends on either of the packages 'tripack' or 'deldir' being installed. For further information, please consult the help page of panel.voronoi. On the chondro data set, plotmap is roughly 5 times faster than plotvoronoi using tripack, and ca. 15 times faster than plotvoronoi using deldir. Package tripack, however, is free only for non-commercial use. Also, it seems that tripack version hang (R running at full CPU power, but not responding nor finishing the calculation) for certain data sets. In this case, mix = TRUE may help.

map.identify calls plotmap and plotvoronoi, respectively and waits for (left) mouse clicks on points. Other mouse clicks end the input.
Unlike \texttt{panel.identify}, the indices returned by \texttt{map.identify} are in the same order as the points were clicked. Also, multiple clicks on the same point are returned as multiple entries with the same index.

\texttt{map.identify} uses option debuglevel similar to \texttt{spc.identify}: debuglevel == 1 will plot the tolerance window if no data point was inside (and additionally labels the point) while debuglevel == 2 will always plot the tolerance window.

The \texttt{map.sel.*} functions offer further interactive selection, see \texttt{map.sel.poly}.

### Value

\texttt{map.identify} returns a vector of row indices into object of the clicked points.

The other functions return a lattice object.

### Author(s)

C. Beleites

### See Also

\texttt{vignette (plotting)}, \texttt{vignette (introduction)}

\texttt{plot}

\texttt{levelplot}

\texttt{trellis.factor.key} for improved color coding of factors

\texttt{hyperSpec options spc.identify map.sel.poly}

\texttt{panel.voronoi}

### Examples

```r
## Not run:
vignette (plotting)
vignette (introduction)

## End(Not run)

levelplot (spc ~ y * x, chondro [,,1003]) # properly rotated
plotmap (chondro [,,1003])

# plot spectra matrix
levelplot (spc ~ .wavelength * t, laser, contour = TRUE, col = "#00000080")
# see also plotmat

plotmap (chondro, clusters ~ x * y)

# Voronoi plots
smpl <- sample (chondro, 300)
plotmap (smpl, clusters ~ x * y)
if (require (tripack))
  plotvoronoi (smpl, clusters ~ x * y)
```
if (require(deldir))
  plotvoronoi(smpl, clusters ~ x * y,
          use.tripack = FALSE)

plotmat

Plot spectra matrix

Description
plots the spectra matrix.

Usage
plotmat(object, y = "row", ylab, col = alois.palette(20), ...,
         contour = FALSE)

Arguments
object hyperSpec object
y character giving the name of the extra data column to label the y axis.
ylab y axis label, defaults to "row" and the label of the extra data column used for
     the y axis, respectively.
col see image
... further parameters for image
contour should contour be called instead of image?

Details
If package plotrix is available, a color legend is plotted to the right. The right margin is set to at
least 5 lines.

Author(s)
Claudia Beleites

See Also
image, contour, levelplot
Examples

plotmat (laser, col = alois.palette (100))

plot (laser, "mat")

plotmat (laser)
plotmat (laser, contour = TRUE, add = TRUE)

## use different y axis labels

plotmat (laser, "t")

plotmat (laser, laser$t / 3600, ylab = "t / h")

---

plotspc  

**Plotting Spectra**  
Plot the spectra of a hyperspec object, i.e. intensity over wavelength. Instead of the intensity values of the spectra matrix summary values calculated from these may be used.

---

**Description**

This is hyperspec’s main plotting function for spectra plots.

Usually, the stacked argument of `plotspc` will do fine, but if you need fine control over the stacking, you may calculate the y offsets yourself.

**Usage**

```r
plotspc(object, wl.range = NULL, wl.index = FALSE, wl.reverse = FALSE,  
spc.nmax = 50, func = NULL, func.args = list(), stacked = NULL,  
stacked.args = list(), add = FALSE, bty = "l", plot.args = list(),  
col = "black", lines.args = list(), xoffset = 0, yoffset = 0,  
nxticks = 10, axis.args = list(), break.args = list(),  
title.args = list(), fill = NULL, fill.col = NULL, border = NA,  
polygon.args = list(), zeroline = list(lty = 2, col = col))
```

`stacked.offsets(x, stacked = TRUE, min.zero = FALSE, add.factor = 0.05,  
add.sum = 0, .spc = NULL)`

**Arguments**

- **object**  
  the hyperspec object

- **wl.range**  
  the wavelength range to be plotted.
  Either a numeric vector or a list of vectors with different wavelength ranges to be plotted separately.
  The values can be either wavelengths or wavelength indices (according to `wl.index`).

- **wl.index**  
  if TRUE, `wl.range` is considered to give column indices into the spectra matrix.
  Defaults to specifying wavelength values rather than indices.
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>wl.reverse</code></td>
<td>if TRUE, the wavelength axis is plotted backwards.</td>
</tr>
<tr>
<td><code>spc.nmax</code></td>
<td>maximal number of spectra to be plotted (to avoid accidentally plotting of large numbers of spectra).</td>
</tr>
<tr>
<td><code>func</code></td>
<td>a function to apply to each wavelength in order to calculate summary spectra such as mean, min, max, etc.</td>
</tr>
<tr>
<td><code>func.args</code></td>
<td>list with further arguments for <code>func</code></td>
</tr>
<tr>
<td><code>stacked</code></td>
<td>if not NULL, a &quot;stacked&quot; plot is produced, see the example. stacked may be TRUE to stack single spectra. A numeric or factor is interpreted as giving the grouping, character is interpreted as the name of the extra data column that holds the groups.</td>
</tr>
<tr>
<td><code>stacked.args</code></td>
<td>a list with further arguments to <code>stacked.offsets</code>.</td>
</tr>
<tr>
<td><code>add</code></td>
<td>if TRUE, the output is added to the existing plot</td>
</tr>
<tr>
<td><code>bty</code></td>
<td>see <code>par</code></td>
</tr>
<tr>
<td><code>plot.args</code></td>
<td>list with further arguments to <code>plot</code></td>
</tr>
<tr>
<td><code>col</code></td>
<td>see <code>par</code>. <code>col</code> might be a vector giving individual colors for the spectra.</td>
</tr>
<tr>
<td><code>lines.args</code></td>
<td>list with further arguments to <code>lines</code>. <code>lines.argstyping</code> defaults to &quot;l&quot;.</td>
</tr>
<tr>
<td><code>xoffset</code></td>
<td>vector with abscissa offsets for each of the <code>wl.range</code> values. If it has one element less than there are <code>wl.range</code>, 0 is padded at the beginning.</td>
</tr>
<tr>
<td></td>
<td>The values are interpreted as the distance along the wavelength axis that the following parts of the spectra are shifted towards the origin. E.g. if <code>wl.range = list(600 ~ 1800, 2800 ~ 6000)</code> and <code>xoffset = 750</code> would result in a reasonable plot. See also the examples.</td>
</tr>
<tr>
<td><code>yoffset</code></td>
<td>ordinate offset values for the spectra. May be offsets to stack the spectra (<code>stacked.offsets</code>). Either one for all spectra, one per spectrum or one per group in stacked.</td>
</tr>
<tr>
<td><code>nxticks</code></td>
<td>hint how many tick marks the abscissa should have.</td>
</tr>
<tr>
<td><code>axis.args</code></td>
<td>list with further arguments for <code>axis</code>. <code>axis.args$x</code> should contain arguments for plotting the abscissa, <code>axis.args$y</code> those for the ordinate (again as lists).</td>
</tr>
<tr>
<td><code>break.args</code></td>
<td>list with arguments for <code>axis.break</code>.</td>
</tr>
<tr>
<td><code>title.args</code></td>
<td>list with further arguments to <code>title</code>. <code>title.args</code> may contain two lists, <code>$x</code>, and <code>$y</code> to set parameters individually for each axis.</td>
</tr>
<tr>
<td><code>fill</code></td>
<td>if not NULL, the area between the specified spectra is filled with color <code>col</code>. The grouping can be given as factor or numeric, or as a character with the name of the extra data column to use. If a group contains more than 2 spectra, the first and the last are used. If TRUE spectra n and nrow (spc) - n build a group.</td>
</tr>
<tr>
<td><code>fill.col</code></td>
<td>character vector with fill color. Defaults to brightened colors from <code>col</code>.</td>
</tr>
<tr>
<td><code>border</code></td>
<td>character vector with border color. You will need to set the line color <code>col</code> to <code>NA</code> in order see the effect.</td>
</tr>
<tr>
<td><code>polygon.args</code></td>
<td>list with further arguments to <code>polygon</code> which draws the filled areas.</td>
</tr>
</tbody>
</table>
zeroline: NA or a list with arguments `abline`, used to plot line(s) marking \( I = 0 \). NA suppresses plotting of the line. The line is by default turned off if `yoffset` is not 0.

\( x \): a hyperSpec object

\( \text{min.zero} \): if TRUE, the lesser of zero and the minimum intensity of the spectrum is used as minimum.

\( \text{add.factor, add.sum} \): proportion and absolute amount of space that should be added.

\( .spc \): for internal use. If given, the ranges are evaluated on \( .spc \). However, this may change in future.

Details
New plots are created by `plot`, but the abscissa and ordinate are drawn separately by `axis`. Also, `title` is called explicitly to set up titles and axis labels. This allows fine-grained customization of the plots.

If package `plotrix` is available, its function `axis.break` is used to produce break marks for cut wavelength axes.
Empty levels of the stacking factor are dropped (as no stacking offset can be calculated in that case.)

Value
`plotspc` invisibly returns a list with

\( x \): the abscissa coordinates of the plotted spectral data points

\( y \): a matrix the ordinate coordinates of the plotted spectral data points

\( \text{wavelengths} \): the wavelengths of the plotted spectral data points

This can be used together with `spc.identify`.

a list containing

\( \text{offsets} \): numeric with the yoffset for each group in stacked groups

\( \text{groups} \): numeric with the group number for each spectrum

\( \text{levels} \): if stacked is a factor, the levels of the groups

Author(s)
C. Beleites

See Also
`plot`, `axis`, `title`, `lines`, `polygon`, `par` for the description of the respective arguments.

`axis.break` for cut marks
See `plot` for some predefined spectra plots such as mean spectrum +/- one standard deviation and the like.
`identify` and `locator` about interaction with plots.

`plotspc`
Examples

```r
plotspc (flu)

## artificial example to show wavelength axis cutting
plotspc (chondro [sample (nrow (chondro), 50)],
  wl.range = list (600 ~ 650, 1000 ~ 1100, 1600 ~ 1700),
  xoffset = c (0, 300, 450))

plotspc (chondro [sample (nrow (chondro), 50)],
  wl.range = list (600 ~ 650, 1000 ~ 1100, 1600 ~ 1700),
  xoffset = c (300, 450))

## some journals publish Raman spectra backwards
plotspc (chondro [sample (nrow (chondro), 50)], wl.reverse = TRUE)

plotspc (laser[(0:4)*20+1,], stacked = TRUE)

plotspc (laser, func = mean_pm_sd,
  col = c(NA, "red", "black"), lines.args = list (lwd = 2),
  fill = c (1, NA, 1),
  fill.col = "yellow", border = "blue",
  polygon.args = list (lty = 2, lwd = 4),
  title.args = list (xlab = expression (lambda[emission] / nm),
      y = list(line = 3.4),
      col.lab = "darkgreen"),
  axis.args = list (x = list (col = "magenta"), y = list (las = 1))
)

mean.pm.sd <- aggregate (chondro, chondro$clusters, mean_pm_sd)
plot (mean.pm.sd, col = matlab.palette (3), fill = ".aggregate", stacked = ".aggregate")

mean.pm.sd <- aggregate (chondro, chondro$clusters, mean_pm_sd)
offset <- stacked.offsets (mean.pm.sd, ".aggregate")
plot (mean.pm.sd, fill.col = matlab.palette (3), fill = ".aggregate",
  stacked = ".aggregate")

plot (aggregate (chondro, chondro$clusters, mean), yoffset = offset$offsets,
  lines.args = list (lty = 2, lwd = 2), add = TRUE)

barb <- do.call (collapse, barbiturates [1:3])
plot (barb, lines.args = list (type = "h"), stacked = TRUE,
  stacked.args = list (add.factor = .2))
```

---

**qplotc**  
*Spectra plotting with ggplot2*
Description

Spectra plotting with ggplot2

Usage

qplotc(object, mapping = aes_string(x = "c", y = "spc"), ..., func = NULL,
func.args = list(), map.pointonly = FALSE)

Arguments

object
hyperSpec object

mapping see geom_point

... handed to geom_point

func function to summarize the wavelengths, if NULL, only the first wavelength is used

func.args arguments to func

map.pointonly if TRUE, mapping will be handed to geom_point instead of ggplot.

Details

These functions are still experimental and may change substantially in future.

Value

a ggplot object

Author(s)

Claudia Beleites

See Also

plotc

ggplotgeom_point

Examples

qplotc (flu)
qplotc (flu) + geom_smooth (method = "lm")
Description
Spectra plotting with ggplot2

Usage
qplotmap(object, mapping = aes_string(x = "x", y = "y", fill = "spc"), ..., 
func = mean, func.args = list(), map.tileonly = FALSE)

Arguments
- object: hyperSpec object
- mapping: see geom_tile
- ...: handed to geom_tile
- func: function to summarize the wavelengths
- func.args: arguments to func
- map.tileonly: if TRUE, mapping will be handed to geom_tile instead of ggplot.

Details
These functions are still experimental and may change substantially in future.
Note that qplotmap will currently produce the wrong scales if x or y are discrete.

Value
a ggplot object

Author(s)
Claudia Beleites

See Also
plotmap
ggplotgeom_tile

Examples
qplotmap (chondro)
qplotmap (chondro) + scale_fill_gradientn (colours = alois.palette ())

## works also with discrete x or y axis:
qplotmap (chondro, mapping = aes (x = x, y = as.factor (y), fill = spc))
qplotmixmap

qplotmap with colour mixing for multivariate overlay

Description

map plot with colour overlay.

Usage

qplotmixmap(object, ...)

Arguments

object hyperSpec object

... handed over to qmixlegend and qmixtile

Value

invisible list with ggplot2 objects map and legend

Author(s)

Claudia Beleites

See Also

qmixtile

Examples

chondro <- chondro - spc.fit.poly.below(chondro)
chondro <- sweep(chondro, 1, apply(chondro, 1, mean), "/")
chondro <- sweep(chondro, 2, apply(chondro, 2, quantile, 0.05), "/")

qplotmixmap(chondro [,c (940, 1002, 1440)],
    purecol = c (colg = "red", Phe = "green", Lipid = "blue"))

qplotmixmap(chondro [,c (940, 1002, 1440)],
    purecol = c (colg = "red", Phe = "green", Lipid = "blue"))
qplotspc

---

**qplotspc**

*Spectra plotting with ggplot2*

### Description

Spectra plotting with ggplot2

### Usage

```r
qplotspc(x, wl.range, ..., mapping = aes_string(x = ".wavelength", y = "spc", group = ".rownames"), spc.nmax = 10, map.lineonly = FALSE)
```

### Arguments

- `x`: hyperSpec object
- `wl.range`: wavelength ranges to plot
- `...`: handed to `geom_line`
- `mapping`: see `geom_line`
- `spc.nmax`: maximum number of spectra to plot
- `map.lineonly`: if TRUE, mapping will be handed to `geom_line` instead of `ggplot`.

### Details

These functions are still experimental and may change substantially in future.

### Value

A `ggplot` object

### Author(s)

Claudia Beleites

### See Also

- `plotspc`
- `ggplot geom_line`

### Examples

```
qplotspc (chondro)
qplotspc (paracetamol, c (2800 - max, min ~ 1800)) + scale_x_reverse (breaks = seq (0, 3200, 400))
qplotspc (aggregate (chondro, chondro$clusters, mean),
  mapping = aes (x = .wavelength, y = spc, colour = clusters)) +
```
facet_grid (clusters ~ .)
qplotspc (aggregate (chondro, chondro$clusters, mean_pm_sd),
  mapping = aes (x = .wavelength, y = spc, colour = clusters, group = .rownames)) +
  facet_grid (clusters ~ .)

---

**rbind.fill.matrix**  
*Bind matrices by row, and fill missing columns with NA*

**Description**

The matrices are bound together using their column names or the column indices (in that order of precedence.) Numeric columns may be converted to character beforehand, e.g. using format. If a matrix doesn’t have colnames, the column number is used.

The matrices are bound together using their column names or the column indices (in that order of precedence.) Numeric columns may be converted to character beforehand, e.g. using format. If a matrix doesn’t have colnames, the column number is used (via `make.names(unique = TRUE)`).

Combine objects by row, filling in missing columns. `rbind` adds a list of data frames filling missing columns with NA.

This is an enhancement to `rbind` which adds in columns that are not present in all inputs, accepts a list of data frames, and operates substantially faster.

**Usage**

```r
rbind.fill.matrix(...)  
rbind.fill.matrix(...)  
rbind.fill(...)```

**Arguments**

...  
data frames/matrices to row bind together

**Details**

Note that this means that a column with name "X1" is merged with the first column of a matrix without name and so on.

Vectors are converted to 1-column matrices prior to `rbind`.

Matrices of factors are not supported. (They are anyways quite inconvenient.) You may convert them first to either numeric or character matrices. If a character matrix is merged with a numeric, the result will be character.

Row names are ignored.

The return matrix will always have column names.
Value

a matrix
a matrix

Author(s)

C. Beleites
C. Beleites

See Also

rbind, cbind, rbind.fill
rbind, cbind, rbind.fill

Examples

A <- matrix(1:4, 2)
B <- matrix(6:11, 2)
A
B
rbind.fill.matrix(A, B)

colnames(A) <- c(3, 1)
A
rbind.fill.matrix(A, B)

A <- matrix(1:4, 2)
B <- matrix(6:11, 2)
A
B
rbind.fill.matrix(A, B)

colnames(A) <- c(3, 1)
A
rbind.fill.matrix(A, B)

rbind.fill.matrix(A, 99)

# rbind.fill(mtcars[,c("mpg", "wt")], mtcars[,c("wt", "cyl")])

read.ENVI.Nicolet Import of ENVI data as hyperSpec object

Description

This function allows ENVI data import as hyperSpec object.
read.ENVI.Nicolet should be a good starting point for writing custom wrappers for read.ENVI that take into account your manufacturer’s special entries in the header file.
Usage

read.ENVI.Nicolet(file = stop("read.ENVI: file name needed"),
headerfile = NULL, header = list(), ..., x = NA, y = NA,
nicolet.correction = FALSE)

read.ENVI(file = stop("read.ENVI: file name needed"), headerfile = NULL,
header = list(), keys.hdr2data = FALSE, x = 0:1, y = x,
wavelength = NULL, label = list())

Arguments

file complete name of the binary file
headerfile name of the ASCII header file. If NULL, the name of the header file is guessed by looking for a second file with the same basename but different suffix as file.
header list with the respective information, see details.
... handed to read.ENVI
x, y vectors of form c(offset, step size) for the position vectors, see details.
nicolet.correction see details
keys.hdr2data determines which fields of the header file should be put into the extra data. Defaults to none.
   To specify certain entries, give character vectors containing the lowercase names of the header file entries.
wavelength, label lists that overwrite the respective information from the ENVI header file. These data is then handed to initialize

Details

Nicolet uses some more keywords in their header file. They are interpreted as follows:

description giving the position of the first spectrum
z plot titles wavelength and intensity axis units, comma separated
pixel size interpreted as x and y step size (specify x = NA and y = NA)

These parameters can be overwritten by giving a list with the respective elements in parameter header.

The values in header line description seem to be microns while the pixel size seems to be in microns. If nicolet.correction is true, the pixel size values (i.e. the step sizes) are multiplied by 1000.

ENVI data usually consists of two files, an ASCII header and a binary data file. The header contains all information necessary for correctly reading the binary file.

I experienced missing header files (or rather: header files without any contents) produced by Bruker Opus’ ENVI export.

In this case the necessary information can be given as a list in parameter header instead. The elements of header are then:
Some more information that is not provided by the ENVI files may be given:

Wavelength axis and axis labels in the respective parameters. For more information, see `initialize`.

The spatial information is by default a sequence from 0 to \( \text{header}D$\)\(\text{samples} - 1\) and \(\text{header}D$\)\(\text{lines} - 1\), respectively. \(x\) and \(y\) give offset of the first spectrum and step size.

Thus, the object’s \(x\) column is: \((0 : \text{header}D$\text{samples} - 1) \times [2] + x [1]\). The \(y\) column is calculated analogously.

Value

a hyperSpec object

Functions

- `read.ENVI.Nicolet`

Author(s)

C. Beleites, testing for the Nicolet files C. Dicko

References

This function was adapted from `read.ENVI`:

read.ini
Read INI files

Description
read.ini reads ini files of the form

Usage
read.ini(con = stop("Connection con needed."), skip = NULL)

Arguments
con connection or file name
skip number of lines to skip before first [section] starts

Details
[section] key = value
into a list.
read.ini sanitizes the element names and tries to convert scalars and comma separated numeric vectors to numeric.

Value
a list with one element per section in the .ini file, each containing a list with elements for the key-value-pairs.

Author(s)
C. Beleites
**Description**

this is a first rough import function for JCAMP-DX spectra.

**Usage**

```r
read.jdx(filename = stop("filename is needed"), encoding = "", header = list(), keys.hdr2data = FALSE, ..., NA.symbols = c("NA", "N/A", "N.A."), collapse.multi = TRUE)
read.jdx.Shimadzu(...)```

**Arguments**

- **filename**: file name and path of the .jdx file
- **encoding**: encoding of the JCAMP-DX file (used by `readLines`)
- **header**: list with manually set header values
- **keys.hdr2data**: index vector indicating which header entries should be tranferred into the extra data. Usually a character vector of labels (lowercase, without and dashes, blanks, underscores). If TRUE, all header entries are read.
- **xtol, ytol**: further parameters handed to the data import function, e.g.
- **NA.symbols**: character vector of text values that should be converted to NA
- **collapse.multi**: should hyperSpec objects from multispectra files be collapsed into one hyper-Spec object (if FALSE, a list of hyperSpec objects is returned).

**Details**

So far, AFFN and PAC formats are supported for simple XYDATA, DATA TABLEs and PEAK TABLEs.

NTUPLES / PAGES are not (yet) supported.

DIF, DUF, DIFDUP and SQZ data formats are not (yet) supported.

**Value**

hyperSpec object
Note

JCAMP-DX support is incomplete and the functions may change without notice. See vignette ("fileio") and the details section.

read.jdx.Shimadzu is now defunct. Please use read.jdx instead.

Author(s)

C. Beleites with contributions by Bryan Hanson

read.mat.Cytospec  Import for Cytospec mat files

Description

These functions allow to import .mat (Matlab V5) files written by Cytospec.

Usage

read.mat.Cytospec(file, keysdata = FALSE, blocks = TRUE)

read.cytomat(...)

Arguments

file  The complete file name (or a connection to) the .mat file.
keys2data  specifies which elements of the Info should be transferred into the extra data
blocks  which blocks should be read? TRUE reads all blocks.
...  read.cytomat for now hands all arguments to read.mat.Cytospec for backwards compatibility.

Details

read.cytomat has been renamed to read.mat.Cytospec and is now deprecated. Use read.mat.Cytospec instead.

Value

hyperSpec object if the file contains a single spectra block, otherwise a list with one hyperSpec object for each block.

Note

This function is an ad-hoc implementation and subject to changes.

Author(s)

C. Beleites
read.spc

**See Also**

R.matlab::readMat

---

**Description**

Import for Thermo Galactic’s spc file format These functions allow to import Thermo Galactic/Grams .spc files.

**Usage**

```r
read.spc(filename, keys.hdr2data = FALSE, keys.hdr2log = FALSE,
          keys.log2data = FALSE, keys.log2log = FALSE, log.txt = TRUE,
          log.bin = FALSE, log.disk = FALSE, hdr = list(), no.object = FALSE)
```

**Arguments**

- `filename` The complete file name of the .spc file.
- `keys.hdr2data`, `keys.hdr2log` character vectors with the names of parameters in the .spc file’s log block (log2xxx) or header (hdr2xxx) that should go into the extra data (yyy2data) of the returned hyperSpec object.
- `log.txt` Should the text part of the .spc file’s log block be read?
- `log.bin, log.disk` Should the normal and on-disk binary parts of the .spc file’s log block be read? If so, they will be put as raw vectors into the hyperSpec object’s log.
- `hdr` A list with fileheader fields that overwrite the settings of actual file’s header. Use with care, and look into the source code for detailed insight on the elements of this list.
- `no.object` If TRUE, a list with wavelengths, spectra, labels, log and data are returned instead of a hyperSpec object. This parameter will likely be subject to change in future - use with care.

**Value**

If the file contains multiple spectra with individual wavelength axes, `read.spc` returns a list of hyperSpec objects. Otherwise the result is a hyperSpec object.

`read.spc.KaiserMap` returns a hyperSpec object with data columns x, y, and z containing the stage position as recorded in the .spc files’ log.
Note

Only a restricted set of test files was available for development. Particularly, the w-planes feature could not be tested.

If you have .spc files that cannot be read with these function, don’t hesitate to contact the package maintainer with your code patch or asking advice.

Author(s)

C. Beleites

References

Source development kit and file format specification of .spc files.

See Also

textio

Examples

```r
## get the sample .spc files from ftirsearch.com (see above)
## Not run:
# single spectrum
spc <- read.spc("BENZENE.SPC")
plot(spc)

# multi-spectra .spc file with common wavelength axis
spc <- read.spc("IG_MULTI.SPC")
spc

# multi-spectra .spc file with individual wavelength axes
spc <- read.spc("BARBITUATES.SPC")
plot(spc [[1]], lines.args = list(type = "h"))
## End(Not run)
```

---

read.spc.Kaiser  read Kaiser .spc files

Description

Import functions for Kaiser Optical Systems .spc files
read.spc.KaiserMap is a wrapper for read.spc.Kaiser with predefined log2data to fetch the stage position for each file.
read.spc.KaiserLowHigh is a wrapper for read.spc.Kaiser for raw data that is saved in separate files for low and high wavenumber range. The wavelength axis holds the pixel numbers, which repeat for low and high wavenumber ranges.
Usage

read.spc.Kaiser(files, ..., glob = TRUE)

read.spc.KaiserMap(files, keys.log2data = NULL, ...)

read.spc.KaiserLowHigh(files = stop("file names needed"), type = c("single", "map"), ..., glob = TRUE)

Arguments

files If glob = TRUE, filename can contain wildcards. Thus all files matching the name pattern in filename can be specified.

glob If TRUE the filename is interpreted as a wildcard containing file name pattern and expanded to all matching file names.

keys.log2data, ... All further arguments are handed over directly to read.spc.

type what kind of measurement was done? If "map", read.spc.KaiserMap is used instead of read.spc.Kaiser.

Details

read.spc.Kaiser imports sets of .spc files written by Kaiser Optical Systems’ Hologram software. It may also serve as an example how to write wrapper functions for read.spc to conveniently import specialized sets of .spc files.

Value

hyperSpec

Examples

### for examples, please see `vignette("fileio", package = "hyperSpec")`.

---

read.spe Import WinSpec SPE file

Description

Import function for WinSpec SPE files (file version 2.5). The calibration data (polynome and calibration data pairs) for x-axis are automatically read and applied to the spectra. Note that the y-calibration data structure is not extracted from the file since it is not saved there by WinSpec and is always empty.
Usage

```r
read.spe(filename, xaxis = "file", acc2avg = F, cts_sec = F,
keys.hdr2data = c("exposure_sec", "LaserWavelen", "accumulCount",
"numFrames", "darkSubtracted"))
```

```r
read.spe.header(filename)
```

```r
spe.showcalpoints(filename, xaxis = "file", acc2avg = F, cts_sec = F)
```

Arguments

- **filename** Name of the SPE file to read data from
- **xaxis** Units of x-axis, e.g. "file", "px", "nm", "energy", "raman", ... read.spe function automatically checks if the x-calibration data are available and uses them (if possible) to reconstruct the xaxis in the selected units.
- **acc2avg** whether to divide the actual data set by the number of accumulations, thus transforming accumulated spectra to averaged spectra. WinSpec does not do this automatically, so the spectral intensity is always proportional to the number of accumulations. The flag @data$averaged is automatically set to TRUE.
- **cts_sec** whether to divide the actual data set by the exposure time, thus going to count per second unit.
- **keys.hdr2data** Which metadata from the file header should be saved to the Data slot of a newly created hyperSpec object

Value

- hyperSpec object
- hdr list with key=value pairs

Functions

- **read.spe.header**: Read only header of a WinSpec SPE file (version 2.5)
- **spe.showcalpoints**: Plot the WinSpec SPE file (version 2.5) and show the calibration points stored inside of it (x-axis calibration)

Author(s)

R. Kiselev, C. Beleites
Import Horiba Labspec exported ASCII files

Description

Read ASCII (.txt) files exported by Horiba's Labspec software (LabRAM spectrometers)

read.txt.Horiba.t reads time series, i.e. .txt files with the time in the first column

Usage

read.txt.Horiba(file, cols = c(spc = "I / a.u.", .wavelength = expression(Delta * tilde(nu)/cm^-1)), header = TRUE, sep = "\t", row.names = NULL, check.names = FALSE, ...)

read.txt.Horiba.xy(file, ...)

read.txt.Horiba.t(file, header = TRUE, sep = "\t", row.names = NULL, check.names = FALSE, ...)

Arguments

file
connection (file name and path) to the .txt file

cols, header, sep, row.names, check.names, ...

further parameters are handed over to read.txt.wide

Details

read.txt.Horiba.xy reads maps, i.e. .txt files where the first two columns give x and y coordinates.

Value

hyperSpec object

Author(s)

C. Beleites
Import and Export of hyperSpec objects

Besides save and load, two general ways to import and export data into hyperSpec objects exist.

Description

Firstly, hyperSpec objects can be imported and exported as ASCII files.

Besides save and load, two general ways to import and export data into hyperSpec objects exist.

Usage

read.txt.long(file = stop("file is required"), cols = list(.wavelength = expression(lambda/nm), spc = "I / a.u."), header = TRUE, ...)

read.txt.wide(file = stop("file is required"), cols = list(spc = "I / a.u.", .wavelength = expression(lambda/nm)), sep = "\t", row.names = NULL, check.names = FALSE, ...)

write.txt.long(object, file = ",", order = c(".rownames", ".wavelength"), na.last = TRUE, decreasing = FALSE, quote = FALSE, sep = "\t", row.names = FALSE, cols = NULL, col.names = TRUE, col.labels = FALSE, append = FALSE, ...)

write.txt.wide(object, file = ",", cols = NULL, quote = FALSE, sep = "\t", row.names = FALSE, col.names = TRUE, header.lines = 1, col.labels = if (header.lines == 1) FALSE else TRUE, append = FALSE, ...)

Arguments

file filename or connection
cols the column names specifying the column order.
For data import, a list with elements colname = label; for export a character vector with the colnames. Use wavelength to specify the wavelengths.
header the file has (shall have) a header line

... arguments handed to read.table and write.table, respectively.
check.names handed to read.table. Make sure this is FALSE, if the column names of the spectra are the wavelength values.
object the hyperSpec object
order which columns should be ordered? order is used as index vector into a data.frame with columns given by cols.
na.last handed to order by write.txt.long.
decreasing logical vector giving the sort order
quote, sep, col.names, row.names

    have their usual meaning (see read.table and write.table), but different default values.

    For file import, row.names should usually be NULL so that the first column becomes a extra data column (as opposed to row names of the extra data).

col.labels

    Should the column labels be used rather than the colnames?

append

    Should the output be appended to an existing file?

header.lines

    Toggle one or two line header (wavelengths in the second header line) for write.txt.wide

Details

A second option is using the package \texttt{R.matlab} which provides the functions \texttt{readMat} and \texttt{writeMat}.

hyperSpec comes with a number of pre-defined functions to import manufacturer specific file formats. For details, see vignette ("file-io").

\texttt{read.spc} imports Thermo Galactic’s .spc file format, and ENVI files may be read using \texttt{read.ENVI}.

These functions are very flexible and provide lots of arguments.

If you use them to read or write manufacturer specific ASCII formats, please consider writing a wrapper function and contributing this function to \texttt{hyperSpec}. An example is in the “flu” vignette (see vignette ("flu", package = "hyperSpec").

Note that R accepts many packed formats for ASCII files, see \texttt{connections}. For .zip files, see \texttt{unzip}.

For further information, see the examples below and the documentation of \texttt{R.matlab}.

Firstly, hyperSpec objects can be imported and exported as ASCII files.

A second option is using the package \texttt{R.matlab} which provides the functions \texttt{readMat} and \texttt{writeMat}.

hyperSpec comes with a number of pre-defined functions to import manufacturer specific file formats. For details, see vignette ("fileio").

\texttt{read.spc} imports Thermo Galactic’s .spc file format, and ENVI files may be read using \texttt{read.ENVI}.

These functions are very flexible and provide lots of arguments.

If you use them to read or write manufacturer specific ASCII formats, please consider writing a wrapper function and contributing this function to \texttt{hyperSpec}. An example is in the “flu” vignette (see vignette ("flu", package = "hyperSpec").

Note that R accepts many packed formats for ASCII files, see \texttt{connections}. For .zip files, see \texttt{unzip}.

For further information, see the examples below, vignette ("fileio") and the documentation of \texttt{R.matlab}.

Author(s)

C. Beleites
See Also

- `read.table` and `write.table`
- `R.matlab` for `.mat` files
- `read.ENVI` for ENVI data
- `read.spc` for `.spc` files

Manufacturer specific file formats:
- `scan.txt.Renishaw`
- `vignette` ("fileio") and [http://hyperspec.r-forge.r-project.org/blob/fileio.pdf](http://hyperspec.r-forge.r-project.org/blob/fileio.pdf), respectively

Examples

```r
## Not run: vignette ("file-io")

## export & import matlab files
if (require(R.matlab)){
  # export to matlab file
  writeMat("test.mat", x = flu[[1]], wavelength = flu@wavelength,
           label = lapply(flu@label, as.character))

  # reading a matlab file
  data <- readMat("test.mat")
  print(data)
  mat <- new("hyperSpec", spc = data$x,
             wavelength = as.numeric(data$wavelength),
             label = data$label[,,1])
}

## ascii export & import

write.txt.long(flu, file = "flu.txt", cols = c(".wavelength", "spc", "c"),
               order = c("c", ".wavelength"),
               decreasing = c(FALSE, TRUE))

read.txt.long(file = "flu.txt", cols = list(.wavelength = expression (lambda / nm),
                                          spc = "I / a.u", c = expression ("/(c, (mg/l))")))

write.txt.wide(flu, file = "flu.txt", cols = c("c", "spc"),
               col.labels = TRUE, header.lines = 2, row.names = TRUE)

write.txt.wide(flu, file = "flu.txt", col.labels = FALSE, row.names = FALSE)

read.txt.wide(file = "flu.txt",
              cols = list(c=expression ("/(c, "mg/l")", spc="I / a.u", .wavelength = "lambda / nm"),
                           header = TRUE)
```
**Description**

Reads Shimadzu GCxGC-qMS - Spectra Files (.txt) as exported by Shimadzu Chrome Solution (v. 2.72) Mass Spectrometer: Shimadzu GCMS-QP 2010 Ultra (www.shimadzu.com)

**Usage**

```r
read.txt.Shimadzu(filename, encoding = "", quiet = TRUE)
```

**Arguments**

- **filename**: file name and path of the .txt file
- **encoding**: encoding of the txt file (used by `readLines`)
- **quiet**: suppress printing of progress

**Value**

list of spectra tables

**Note**

This is a first rough import function and the functions may change without notice.

**Author(s)**

Bjoern Egert

**rmmvnorm**

*Multivariate normal random numbers*

**Description**

Interface functions to use `rmvnorm` for `hyperSpec-class` objects.
mmvnorm

Usage

rmrnvmn0r(n, mean, sigma)

## S4 method for signature 'numeric, hyperSpec, matrix'
rmrnvmn0r(n, mean, sigma)

## S4 method for signature 'numeric, hyperSpec, array'
rmrnvmn0r(n, mean, sigma)

## S4 method for signature 'numeric, matrix, matrix'
rmrnvmn0r(n, mean, sigma)

## S4 method for signature 'numeric, matrix, array'
rmrnvmn0r(n, mean, sigma)

Arguments

n vector giving the number of cases to generate for each group
mean matrix with mean cases in rows
sigma common covariance matrix or array (ncol(mean) x ncol(mean) x nrow(mean))
with individual covariance matrices for the groups.

Details

The mvtnorm method for hyperSpec objects supports producing multivariate normal data for groups with different mean but common covariance matrix, see the examples.

See Also

rmvnorm
cov and pooled.cov about calculating covariance of hyperSpec objects.

Examples

## multiple groups, common covariance matrix

if (require("mvtnorm")){
    pcov <- pooled.cov (chondro, chondro$clusters)
    rnd <- rmmvnorm (rep (10, 3), mean = pcov$mean, sigma = pcov$COV)
    plot (rnd, col = rnd$.group)
}
Description

Random Samples and Permutations Take a sample of the specified size from the elements of x with or without replacement.

`isample` returns an vector of indices, `sample` returns the corresponding hyperSpec object.

Usage

```r
## S4 method for signature 'hyperSpec'
sample(x, size = nrow(x), replace = FALSE, prob = NULL)

isample(x, size = nrow(x), replace = FALSE, prob = NULL)

## S4 method for signature 'data.frame'
sample(x, size = nrow(x), replace = FALSE, prob = NULL, drop = FALSE)

## S4 method for signature 'matrix'
sample(x, size = nrow(x), replace = FALSE, prob = NULL, drop = FALSE)
```

Arguments

- `x` The hyperSpec object, data.frame or matrix to sample from/to sample from
- `size` positive integer giving the number of spectra (rows) to choose.
- `replace` Should sampling be with replacement?
- `prob` A vector of probability weights for obtaining the elements of the vector being sampled.
- `drop` see `drop`: by default, do not drop dimensions of the result

Value

- a hyperSpec object, data.frame or matrix with size rows for `sample`, and an integer vector for `isample` that is suitable for indexing (into the spectra) of x.
- vector with indices suitable for row-indexing x

Author(s)

C. Beleites
See Also

sample

Examples

sample (flu, 3)

plot (flu, col = "darkgray")
plot (sample (flu, 3), col = "red", add = TRUE)

plot (flu, col = "darkgray")
plot (sample (flu, 3, replace = TRUE), col = "#0000FF80", add = TRUE,
     lines.args = list(lwd = 2));
isample (flu, 3)
isample (flu, 3, replace = TRUE)
isample (flu, 8, replace = TRUE)
sample (cars, 2)
sample (matrix (1:24, 6), 2)

scale.hyperSpec-method

Center and scale hyperSpec object

Description

link[base]{scale}s the spectra matrix. scale (x, scale = FALSE) centers the data.

Usage

## S4 method for signature 'hyperSpec'
scale(x, center = TRUE, scale = TRUE)

Arguments

x

the hyperSpec object
center

if TRUE, the data is centered to colMeans (x), FALSE suppresses centering. Alternatively, an object that can be converted to numeric of length nwl (x) by as.matrix (e.g. hyperSpec object containing 1 spectrum) can specify the center spectrum.
scale

if TRUE, the data is scaled to have unit variance at each wavelength, FALSE suppresses scaling. Alternatively, an object that can be converted to numeric of length nwl (x) by as.matrix (e.g. hyperSpec object containing 1 spectrum) can specify the center spectrum.
Details

Package scale provides a fast alternative for base::scale

Value

the centered & scaled hyperspec object

Author(s)

C. Beleites

See Also

scale
package scale.

Examples

## mean center & variance scale
tmp <- scale (chondro)
plot (tmp, "spcmeansd")
plot (sample (tmp, 5), add = TRUE, col = 2)

## mean center only
tmp <- scale (chondro, scale = FALSE)
plot (tmp, "spcmeansd")
plot (sample (tmp, 5), add = TRUE, col = 2)

## custom center
tmp <- sweep (chondro, 1, mean, `/`
plot (tmp, "spcmeansd")
tmp <- scale (tmp, center = quantile (tmp, .05), scale = FALSE)

Description

Import Raman Spectra/Maps from Andor Cameras/Solis ASCII files

Usage

scan.asc.Andor(file = stop("filename or connection needed"), ..., quiet = TRUE, dec = ".", sep = ",")
Arguments

file filename or connection to ASCII file
..., quiet, dec, sep
handed to scan

Details

scan.asc.Andor reads Andor Solis ASCII (.asc) files where the first column gives the wavelength axes and the other columns the spectra.

Value

a hyperSpec object

Author(s)

Claudia Beleites

See Also

vignette ("fileio") for more information on file import and
options for details on options.

---

scan.txt.Renishaw

import Raman measurements from Renishaw ASCII-files import Raman measurements from Renishaw (possibly compressed).txt file.

Description

The file may be of any file type that can be read by gzfile (i.e. text, or zipped by gzip, bzip2, xz or lzma). .zip zipped files need to be read using scan.zip.Renishaw.

Usage

scan.txt.Renishaw(file = stop("file is required"), data = "xyspc",
                 nlines = 0, nspc = NULL)

scan.zip.Renishaw(file = stop("filename is required"),
                   txt.file = sub("[.]zip", ".txt", basename(file)), ...)
Arguments

file    file name or connection

data    type of file, one of "spc", "xyspc", "zspc", "depth", "ts", see details.
nlines  number of lines to read in each chunk, if 0 or less read whole file at once.
         nlines must cover at least one complete spectrum, i.e. nlines must be at least the number of data points per spectrum. Reasonable values start at 1e6.
nspc    number of spectra in the file
txt.file name of the .txt file in the .zip archive. Defaults to zip file's name with suffix .txt instead of .zip

...  Arguments for scan.txt.Renishaw

Details

Renishaw .wxd files are converted to .txt ASCII files by their batch converter. They come in a "long" format with columns (y x l time l z)? wavelength intensity. The first columns depend on the data type.

The corresponding possibilities for the data argument are:

<table>
<thead>
<tr>
<th>data</th>
<th>columns</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;spc&quot;</td>
<td>wl int</td>
<td>single spectrum</td>
</tr>
<tr>
<td>&quot;zspc&quot;, &quot;depth&quot;</td>
<td>z wl int</td>
<td>depth profile</td>
</tr>
<tr>
<td>&quot;ts&quot;</td>
<td>t wl int</td>
<td>time series</td>
</tr>
<tr>
<td>&quot;xyspc&quot;</td>
<td>y x wl int</td>
<td>2d map</td>
</tr>
</tbody>
</table>

This function allows reading very large ASCII files, but it does not work on files with missing values (NAs are allowed).

If the file is so large that it should be read in chunks and nspc is not given, scan.txt.Renishaw tries to guess it by using wc (if installed).

Value

the hyperSpec object

Author(s)

C. Beleites

See Also

read.txt.long, read.txt.wide, scan
Description

Import Raman Spectra/Maps from Witec Instrument via ASCII files

Usage

```r
scan.txt.Witec(file = stop("filename or connection needed"),
    points.per.line = NULL, lines.per.image = NULL, nwl = NULL,
    remove.zerospc = TRUE, type = c("single", "map"), hdr.label = FALSE,
    hdr.units = FALSE, ..., quiet = TRUE)
```

```r
scan.dat.Witec(filex = stop("filename or connection needed"),
    filey = sub("-x", ",y", filex), points.per.line = NULL,
    lines.per.image = NULL, type = c("single", "map"), ..., quiet = hy.getOption("debuglevel") < 1L)
```

```r
scan.txt.Witec.Graph(headerfile = stop("filename or connection needed"),
    filex = gsub("Header", "X-Axis", headerfile), filey = gsub("Header", "Y-Axis", headerfile), type = c("single", "map"), ..., quiet = TRUE)
```

Arguments

- **file**: filename or connection to ASCII file
- **points.per.line**: number of spectra in x direction of the map
- **lines.per.image**: number of spectra in y direction
- **nwl**: is deprecated and will be removed soon. Number of wavelengths is calculated automatically.
- **remove.zerospc**: is deprecated and will be removed soon. Use `hy.setOptions` (file.remove.emptyspc = TRUE) instead.
- **type**: type of spectra: single for single spectra (including time series), map for imaging data.
- **hdr.label**: WITec Project exports the spectra names (contain information of map position or number of spectra) within the file.
- **hdr.units**: WITec Project exports the spectra units within the file.
- **..., quiet**: handed to `scan`
- **filex**: filename wavelength axis file
- **filey**: filename intensity file
- **headerfile**: filename or connection to ASCII file with header information
**seq.hyperSpec**

**Details**

scan.txt.Witec reads Witec ASCII files where the first column gives the wavelength axes and the other columns the spectra. scan.dat.Witec reads Witec’s ASCII exported data which comes in separate files with x and y data.

**Value**

a hyperSpec object

**Author(s)**

Claudia Beleites and Marcel Dahms

**See Also**

vignette ("fileio") for more information on file import and options for details on options.

---

**seq.hyperSpec**

Sequence generation along spectra or wavelengths This function generates sequences along the spectra (rows) or wavelengths of hyperSpec objects.

**Description**

Note that \texttt{wlRi} generates sequences of indices along the wavelength axis.

**Usage**

```r
## S3 method for class 'hyperSpec'
seq(x, from = 1, to = nrow(x), ..., index = FALSE)
```

**Arguments**

- \texttt{x} the hyperSpec object
- \texttt{from, to} arguments handed to \texttt{seq.int}
- \texttt{...} arguments for \texttt{seq}, namely by, length.out
- \texttt{index} should a vector with indices be returned rather than a hyperSpec object?

**Details**

seq had to be implemented as S3 method as the generic has only \ldots arguments (on which no dispatch with differing types is possible).

\texttt{seq_along} is not generic, but returns a sequence of the length of the object. As hyperSpec provides a Method \texttt{length}, it can be used. The result is a sequence of indices for the spectra.
Value

a numeric or hyperSpec object, depending on index.

Author(s)

C. Beleites

See Also

wl2i to construct sequences of wavelength indices.

Examples

seq (flu, index = TRUE)
seq_along (flu)
seq (flu, length.out = 3, index = TRUE) # return value is numeric, not integer!
seq (flu, by = 2, index = TRUE) # return value is numeric, not integer!

plot (flu, col = "darkgray")
plot (seq (flu, by = 2), add = TRUE, col= "red")
plot (seq (flu, length.out = 2), add = TRUE, col= "blue")

spc.bin

Wavelength Binning In order to reduce the spectral resolution and thus gain signal to noise ratio or to reduce the dimensionality of the spectral data set, the spectral resolution can be reduced.

Description

The mean of every by data points in the spectra is calculated.

Usage

spc.bin(spc, by = stop("reduction factor needed"), na.rm = TRUE, ...)

Arguments

spc the hyperSpec object
by reduction factor
na.rm decides about the treatment of NAs:
if FALSE or 0, the binning is done using na.rm = FALSE
if TRUE or 1, the binning is done using na.rm = TRUE
if 2, the binning is done using na.rm = FALSE, and resulting NAs are corrected with mean(...{}), na.rm = TRUE).
...
ignored
Details

Using na.rm = TRUE always takes about twice as long as na.rm = FALSE.

If the spectra matrix does not contain too many NAs, na.rm = 2 is faster than na.rm = TRUE.

Value

A hyperspec object with ceiling (nwl (spc) / by) data points per spectrum.

Author(s)

C. Beleites

Examples

```r
spc <- spc.bin (flu, 5)

plot (flu[1,425:475])
plot (spc[1,425:475], add = TRUE, col = "blue")

nwl (flu)
nwl (spc)
```

---

**spc.fit.poly**

**Polynomial Baseline Fitting**

These functions fit polynomial baselines.

**Description**

Both functions fit polynomials to be used as baselines. If apply.to is NULL, a hyperSpec object with the polynomial coefficients is returned, otherwise the polynomials are evaluated on the spectral range of apply.to.

spc.fit.poly below tries to fit the baseline on appropriate spectral ranges of the spectra in fit.to. For details, see the vignette ("baseline").

**Usage**

```r
spc.fit.poly(fit.to, apply.to = NULL, poly.order = 1)

spc.fit.poly.below(fit.to, apply.to = fit.to, poly.order = 1,
    npts.min = NULL, noise = 0)
```
spc.fit.poly

Arguments

- `fit.to` hyperSpec object on which the baselines are fitted
- `apply.to` hyperSpec object on which the baselines are evaluated. If NULL, a hyperSpec object containing the polynomial coefficients rather than evaluated baselines is returned.
- `poly.order` order of the polynomial to be used
- `npts.min` minimal number of points used for fitting the polynomial
- `noise` noise level to be considered during the fit. It may be given as one value for all the spectra, or for each spectrum separately.

Details

spc.fit.poly calculates the least squares fit of order `poly.order` to the complete spectra given in `fit.to`. Thus `fit.to` needs to be cut appropriately.

Value

hyperSpec object containing the baselines in the spectra matrix, either as polynomial coefficients or as polynomials evaluated on the spectral range of `apply.to`

Author(s)

C. Beleites

See Also

vignette("baseline", package = "hyperSpec")

Examples

```r
# Not run: vignette("baseline", package = "hyperSpec")

spc <- chondro[1:10]
baselines <- spc.fit.poly(spc[, , c(625 - 640, 1785 - 1800)], spc)
plot(spc - baselines)

baselines <- spc.fit.poly.below(spc)
plot(spc - baselines)
```
**spc.identify**

Identifying Spectra and Spectral Data Points This function allows to identify the spectrum and the wavelength of a point in a plot produced by `plotspc`.

### Description

This function first finds the spectrum with a point closest to the clicked position (see `locator`). The distance to the clicked point is evaluated relative to the size of the tolerance window.

### Usage

```
spc.identify(x, y = NULL, wavelengths = NULL, ispc = NULL, 
tol.wl = diff(range(x))/200, tol.spc = diff(range(y))/50, 
point.fn = spc.point.max, formatter = spc.label.default, ..., cex = 0.7, 
adj = c(0, 0.5), srt = 90, warn = TRUE)
```

- `spc.point.max(wl, spc, wlcick)`
- `spc.point.default(wl, spc, wlcick)`
- `spc.point.min(wl, spc, wlcick)`
- `spc.point.sqr(wl, spc, wlcick, delta = 1L)`
- `spc.label.default(ispc, wl, spc, digits = 3)`
- `spc.label.wlonly(ispc, wl, spc, digits = 3)`

### Arguments

- `x` either the abscissa coordinates or the list returned by `plotspc`
- `y` the ordinate values. Giving `y` will override any values from `x$y`.
- `wavelengths` the wavelengths for the data points. Giving `wavelengths` will override any values from `x$wavelengths`.
- `ispc` if a selection of spectra was plotted, their indices can be given in `ispc`. In this case `ispc[i]` is returned rather than `i`.
- `tol.wl`, `tol.spc` tolerance in wavelength and spectral intensity to search around the clicked point. See details.
- `point.fn` function `(wl, spc, wlcick)` to determine the actual point to label, see details.
- `formatter` function `(i, wl, spc)` that produces the labels. If `NULL`, no labels are displayed.
- `...` passed to `text` in order to produce the labels
Should the user be warned if no point is in the considered window? In addition, see the discussion of option debuglevel in the details. If FALSE, the resulting data.frame will have a row of NAs instead.

wl
the wavelength to label

spc
the intensity to label

wlclick
the clicked wavelength

delta
spc.point.sqr fits the parabola in the window wlclick ± delta points.

digits
how many digits of the wavelength should be displayed?

Details

In a second step, maxNfn searches for the actual point to label within the specified wavelength window of that spectrum. This allows to label maxima (or minima) without demanding too precise clicks. Currently, the following functions to determine the precise point:

- spc.point.default uses the clicked wavelength together with its spectral intensity
- spc.point.max the point with the highest intensity in the wavelength window
- spc.point.min the point with the lowest intensity in the wavelength window
- spc.point.sqr maximum of a parabola fit through the point with highest intensity and the two surrounding points

pointNfn is called with the arguments wl containing the considered wavelength window, spc the respective intensities of the closest spectrum, and wlclick the wavelength that was clicked. They return a vector of two elements (wavelength and intensity).

As a last step, a label for the point produced by formatter and plotted using text. Currently, the following formatters are available:

- spc.label.default spectrum number, wavelength
- spc.label.wlonly wavelength

formatter functions receive the number of the spectrum ispc, the wavelength wl, and the spectral intensity spc and produce a character variable suitable for labelling. The predefined formatters surround the label text by spaces in order to easily have an appropriate offset from the point of the spectrum.

The warning issued if no spectral point is inside the tolerance window may be switched of by warn = FALSE. In that case, the click will produce a row of NAs in the resulting data.frame.

spc.identify uses option debuglevel to determine whether debugging output should be produced. debuglevel == 2 will plot the tolerance window for every clicked point, debuglevel == 1 will plot the tolerance window only if no data point was inside. See hyperSpec options for details about retrieving and setting options.

You may want to adjust the plot's ylim to ensure that the labels are not clipped. As a dirty shortcut, xpd = NA may help.
Value

a data.frame with columns

ispc spectra indices of the identified points, i.e. the rows of the hyperSpec object that was plotted. If ispc is given, ispc [i] is returned rather than i.

wavelengths the wavelengths of the identified points

spc the intensities of the identified points

Author(s)

C. Beleites

See Also

locator, plotspc, hyperSpec options

Examples

if (interactive ()){
  ispc <- sample (nrow (laser), 10)
  ispc

  identified <- spc.identify (plotspc (laser[ispc]))

  ## convert to the "real" spectra indices
  ispc [identified$ispc]
  identified$wl
  identified$spc

  ## allow the labels to be plotted into the plot margin
  spc.identify (plotspc (laser[ispc]), ispc = ispc, xpd = NA)

  spc.identify (plotspc (paracetamol, xoffset = 1100, 
                  wl.range = c (600 ~ 1700, 2900 ~ 3150)), 
                  formatter = spc.label.wlonly)

  ## looking for minima
  spc.identify (plot (-paracetamol, wl.reverse = TRUE), 
                  point.fn = spc.point.min, adj = c (1, 0.5))
}

spc.loess

*loess smoothing interpolation for spectra* Spectra can be smoothed and interpolated on a new wavelength axis using loess.

Description

Applying loess to each of the spectra, an interpolation onto a new wavelength axis is performed. At the same time, the spectra are smoothed in order to increase the signal : noise ratio. See loess and loess.control on the parameters that control the amount of smoothing.

Usage

```r
spc.loess(spc, newx, enp.target = nwl(spc)/4, surface = "direct", ...)
```

Arguments

- `spc` the hyperSpec object
- `newx` wavelength axis to interpolate on
- `enp.target`, `surface`, ...
  - parameters for loess and loess.control.

Value

a new hyperSpec object.

Author(s)

C. Beleites

See Also

loess, loess.control

Examples

```r
plot (flu, col = "darkgray")
plot (spc.loess(flu, seq (420, 470, 5)), add = TRUE, col = "red")

flu [[3, ]] <- NA_real_
smooth <- spc.loess(flu, seq (420, 470, 5))
smooth [[, ]]
plot (smooth, add = TRUE, col = "blue")
```
spc.NA.linapprox

**Description**
Replace NAs in the spectra matrix by linear interpolation.

**Usage**
```
spc.NA.linapprox(spc, neighbours = 1, ...)
```

**Arguments**
- `spc`: hyperSpec object with spectra matrix containing NAs
- `neighbours`: how many neighbour data points should be used to fit the line
- `...`: ignored

**Value**
hyperSpec object

**Author(s)**
Claudia Beleites

spc.rubberband

**Description**
Rubberband baseline

**Usage**
```
spc.rubberband(spc, ..., upper = FALSE, noise = 0, spline = TRUE)
```

**Arguments**
- `spc`: hyperSpec object
- `...`: further parameters handed to `smooth.spline`
- `upper`: logical indicating whether the lower or upper part of the hull should be used
- `noise`: noise level to be taken into account
- `spline`: logical indicating whether the baseline should be an interpolating spline through the support points or piecewise linear.
Details
Baseline with support points determined from a convex hull of the spectrum.
Use debuglevel \( \geq 1 \) to obtain debug plots, either directly via function argument or by setting hyperSpec's debuglevel option.

Value
hyperSpec object containing the baselines

Note
This function is still experimental

Author(s)
Claudia Beleites

See Also
spc.fit.poly, spc.fit.poly.below
vignette ("baseline")
hy.setOptions

Examples
plot (paracetamol [, , 175 \sim 1800])
bl <- spc.rubberband (paracetamol [, , 175 \sim 1800], noise = 300, df = 20)
plot (bl, add = TRUE, col = 2)
plot (paracetamol [, , 175 \sim 1800] - bl)

---

spc.smooth.spline  Spectral smoothing by splines

Description
Smoothing splines

Usage
spc.smooth.spline(spc, newx = wl(spc), ...)

Arguments

\begin{itemize}
\item \textbf{spc} \quad hyperSpec object
\item \textbf{newx} \quad wavelength axis to interpolate on
\item \ldots \quad further parameters handed to \texttt{smooth.spline}
\end{itemize}
Details
Spectral smoothing by splines

Value
hyperSpec object containing smoothed spectra

Note
This function is still experimental

Author(s)
Claudia Beleites

See Also
spc.loess
smooth.spline

Examples
p <- paracetamol [,2200 - max]
plot (p, col = "gray")
smooth <- spc.smooth.spline (p [, c (2200 - 2400, 2500 - 2825, 3150 - max)],
wl (paracetamol [, , 2200 - max]),
df = 4, spar = 1)
plot (smooth, col = "red", add = TRUE)
plot (p - smooth)

---

split

Split a hyperSpec object according to groups split divides the
hyperSpec object into a list of hyperSpec objects according to the
groups given by f.

Description
The hyperSpec objects in the list may be bound together again by bind ("r", list_of_hyperSpec_objects).

Usage
## S4 method for signature 'hyperSpec'
split(x, f, drop = TRUE)
subset

Arguments
- `x`: the hyperSpec object
- `f`: a factor giving the grouping (or a variable that can be converted into a factor by `as.factor`)
- `drop`: if TRUE, levels off that do not occur are dropped.

Value
A list of hyperSpec objects.

Author(s)
C. Beleites

See Also
- `split`

Examples

```r
dist <- pearson.dist (chondro[[]])
dend <- hclust (dist, method = "ward")
z <- cutree (dend, h = 0.15)
clusters <- split (chondro, z)
length (clusters)

# difference in cluster mean spectra
plot (apply (clusters[[2]], 2, mean) - apply (clusters[[1]], 2, mean))
```

subset

Description
subset for hyperSpec object

Usage

```r
## S4 method for signature 'hyperSpec'
subset(x, ...)
```

Arguments
- `x`: hyperSpec object
- `...`: handed to `subset (data.frame method)`
Summary

Value
hyperSpec object containing the respective subset of spectra.

Author(s)
Claudia Beleites

See Also
subset

---

Description
all, any,

Usage
## S4 method for signature 'hyperSpec'
Summary(x, ..., na.rm = FALSE)

## S4 method for signature 'hyperSpec'
is.na(x)

Arguments
x        hyperSpec object
...      further objects
na.rm    logical indicating whether missing values should be removed

Details
sum, prod,
min, max,
range, and
is.na
for hyperSpec objects.
All these functions work on the spectra matrix.

Value
sum, prod, min, max, and range return a numeric, all, any, and is.na a logical.
See Also

Summary for the base summary functions.

all.equal and isTRUE

Examples

range (flu)

is.na (flu [, , 405 - 410]);

sweep (flu)

is.na (flu [, , 405 - 410]);

sweep

Sweep Summary Statistic out of an hyperSpec Object

sweep for hyperSpec objects.

Description

Calls sweep for the spectra matrix.

Usage

## S4 method for signature 'hyperSpec'

sweep(x, MARGIN, STATS, FUN = "-", check.margin = TRUE, ...)

Arguments

x a hyperSpec object.

MARGIN direction of the spectra matrix that STATS goes along.

STATS the summary statistic to sweep out. Either a vector or a hyperSpec object.

hyperSpec offers a non-standard convenience function: if STATS is a function, this function is applied first (with the same MARGIN) to compute the statistic. However, no further arguments to the apply function can be given. See the examples.

FUN the function to do the sweeping, e.g. '-' or '/'.

check.margin If TRUE (the default), warn if the length or dimensions of STATS do not match the specified dimensions of x. Set to FALSE for a small speed gain when you know that dimensions match.

... further arguments for FUN

Details

sweep is useful for some spectra preprocessing, like offset correction, substraction of background spectra, and normalization of the spectra.
Value

A hyperSpec object.

Author(s)

C. Beleites

See Also

sweep

Examples

```r
## Substract the background / slide / blank spectrum
# the example data does not have spectra of the empty slide,
# so instead the overall composition of the sample is substracted
background <- apply(chondro, 2, quantile, probs = 0.05)
corrected <- sweep(chondro, 2, background, "-")
plot(corrected, "spcprctl5")

## Offset correction
offsets <- apply(chondro, 1, min)
corrected <- sweep(chondro, 1, offsets, "-"")
plot(corrected, "spcprctl5")

## Min-max normalization (on max amide I)
# the minimum is set to zero by the offset correction.
factor <- apply(corrected, 1, max)
mm.corrected <- sweep(corrected, 1, factor, "/")
plot(mm.corrected, "spcprctl5")

## convenience: give function to compute STATS:
mm.corrected2 <- sweep(corrected, 1, max, "/")
plot(mm.corrected2)

## checking
stopifnot(all(mm.corrected2 == mm.corrected))
```
Usage

trellis.factor.key(f, levelplot.args = list())

Arguments

  f                the factor that will be color-coded
  levelplot.args   a list with levelplot arguments

Value

the modified list with levelplot arguments.

Author(s)

C. Beleites

See Also

levelplot

Examples

chondro$z <- factor(rep(c("a", "a", "d", "c"),
                     length.out = nrow(chondro)),
                     levels = letters[1:4])

str(trellis.factor.key(chondro$z))

plotmap(chondro, z ~ x * y)

## switch off using trellis.factor.key:
## note that the factor levels are collapsed to c(1, 2, 3) rather than
## c(1, 3, 4)
plotmap(chondro, z ~ x * y, transform.factor = FALSE)

plotmap(chondro, z ~ x * y,
        col.regions = c("gray", "red", "blue", "dark green"))

Description

vandermonde generates van der Monde matrices, the hyperSpec method generates a hyperSpec object containing the van der Monde matrix of the wavelengths of a hyperSpec object.
Usage

vanderMonde(x, order, ...)

## S4 method for signature 'hyperSpec'
vanderMonde(x, order, ..., normalize.wl = normalize01)

Arguments

- **x**: object to evaluate the polynomial on
- **order**: of the polynomial
- **...**: hyperSpec method: further arguments to decomposition
- **normalize.wl**: function to transform the wavelengths before evaluating the polynomial (or other function). normalize01 maps the wavelength range to the interval [0, 1]. Use I to turn off.

Details

It is often numerically preferable to map \(wl\) (\(x\)) to \([0, 1]\), see the example.

Value

van der Monde matrix

hyperSpec method: hyperSpec object containing van der Monde matrix as spectra and an additional column ".vdm.order" giving the order of each spectrum (term).

Author(s)

C. Beleites

See Also

- **wl.eval** for calculating arbitrary functions of the wavelength,
- **normalize01**

Examples

plot (vanderMonde (flu, 2))
plot (vanderMonde (flu, 2, normalize.wl = I))
wc

wc word count of ASCII files

Description

wc uses the system command wc

Usage

wc(file, flags = c("lines", "words", "bytes"))

Arguments

file the file name or pattern
flags the parameters to count, character vector with the long form of the parameters

Value

data.frame with the counts and file names, or NULL if wc is not available

Author(s)

C. Beleites

wl

Getting and Setting the Wavelength Axis

wl returns the wavelength axis, wl<- sets it.

Description

The wavelength axis of a hyperspec object can be retrieved and replaced with wl and wl<-, respectively.

Usage

wl(x)

wl(x, label=NULL, digits=6) <- value

Arguments

x a hyperspec object
label The label for the new wavelength axis. See initialize for details.
digits handed to signif. See details.
value either a numeric containing the new wavelength vector, or a list with value$wl containing the new wavelength vector and value$label holding the corresponding label.
Details

When the wavelength axis is replaced, the colnames of `@data$spc` are replaced by the rounded new wavelengths. `digits` specifies the how many significant digits should be used.

There are two ways to set the label of the new wavelength axis, see the examples. If no label is given, a warning will be issued.

Value

- a numeric vector
- `hyperSpec` object

Note

`wl` always sets the complete wavelength axis, without changing the columns of the spectra matrix. If you rather want to cut the spectral range, use `[`, for interpolation along the spectral axis see `spc.loess` and for spectral binning `spc.bin`.

Author(s)

C. Beleites

See Also

- `signif`
- cutting the spectral range: `[`
- interpolation along the spectral axis: `spc.loess`
- spectral binning: `spc.bin`

Examples

```r
wl <- laser

# convert from wavelength to frequency
plot(laser)
wl <- laser, "f / Hz") <- 2.998e8 * wl (laser) * 1e9
plot(laser)

# convert from Raman shift to wavelength
# excitation was at 785 nm
plot(chondro [1])
wl (chondro) <- list(wl = 1e7 / (1e7/785 - wl (chondro)), label = expression (lambda / mm))
plot(chondro [1])
```
wl.eval

Evaluate function on wavelengths of hyperSpec object

Description

This is useful for generating certain types of baseline "reference spectra".

Usage

wl.eval(x, ..., normalize.wl = I)

Arguments

x hyperSpec object

... hyperSpec method: expressions to be evaluated

normalize.wl function to transform the wavelengths before evaluating the polynomial (or other function). Use normalize01 to map the wavelength range to the interval [0, 1].

Value

hyperSpec object containing one spectrum for each expression

Author(s)

C. Beleites

See Also

vanderMonde for polynomials,
normalize01 to normalize the wavenumbers before evaluating the function

Examples

plot (wl.eval (laser, exp = function (x) exp (-x)))
wl2i

Conversion between Wavelength and Spectra Matrix Column Index

wl2i returns the column indices for the spectra matrix for the given wavelengths. i2wl converts column indices into wavelengths.

Description

If wavelength is numeric, each of its elements is converted to the respective index. Values outside the range of x@wavelength become NA.

Usage

wl2i(x, wavelength = stop("wavelengths are required."))
i2wl(x, i)

Arguments

x a hyperSpec object
wavelength the wavelengths to be converted into column indices, either numeric or a formula, see details.
i the column indices into the spectra matrix for which the wavelength is to be computed

Details

If the range is given as a formula (i.e. start ~ end, a sequence index corresponding to start : index corresponding to end is returned. If the wavelengths are not ordered, that may lead to chaos. In this case, call orderwl first.

Two special variables can be used: min and max, corresponding to the lowest and highest wavelength of x, respectively.

start and end may be complex numbers. The resulting index for a complex x is then index (Re (x)) + Im (x)

Value

A numeric containing the resulting indices for wl2i
i2wl returns a numeric with the wavelengths

Author(s)

C. Beleites
Examples

flu
wl2i (flu, 405 : 407)
wl2i (flu, 405 - 407)

## beginning of the spectrum to 407 nm
wl2i (flu, min - 407)

## 2 data points from the beginning of the spectrum to 407 nm
wl2i (flu, min + 2i - 407)

## the first 3 data points
wl2i (flu, min - min + 2i)

## from 490 nm to end of the spectrum
wl2i (flu, 490 - max)

## the last 8 data points
wl2i (flu, max - 7i - max)

## get 450 nm +/- 3 data points
wl2i (flu, 450 - 3i - 450 + 3i)

wl2i (flu, 300 : 400) ## all NA:
wl2i (flu, 600 - 700) ## NULL: completely outside flu's wavelength range

i2wl (chondro, 17:20)

---

wlconv Convert different wavelength units

Description

The following units can be converted into each other: \( nm, \text{cm}^{-1}, eV, \text{THz} \) and Raman shift

Usage

wlconv(points, src, dst, laser = NULL)

nm2raman(x, laser)

nm2invcm(x, ...)

nm2ev(x, ...)

nm2freq(x, ...)
invcm2raman(x, laser)
invcm2nm(x, ...)
invcm2ev(x, ...)
invcm2freq(x, ...)
raman2invcm(x, laser)
raman2nm(x, laser)
raman2ev(x, laser)
raman2freq(x, laser)
ev2raman(x, laser)
ev2invcm(x, ...)
ev2nm(x, ...)
ev2freq(x, ...)
freq2nm(x, ...)
freq2invcm(x, ...)
freq2ev(x, ...)
freq2raman(x, laser)

Arguments

points   data for conversion
src      source unit
dst      destination unit
laser    laser wavelength (required for work with Raman shift)
x        wavelength points for conversion
...      ignored

Functions

• nm2raman: conversion nanometers -> Raman shift (relative wavenumber)
• nm2invcm: conversion nanometers -> inverse cm (absolute wavenumber)
• nm2ev: conversion nanometers -> electronvolt
• nm2freq: conversion \texttt{nm} \rightarrow \textit{frequency in THz}
• invcm2raman: conversion \textit{inverse cm (absolute wavenumber)} \rightarrow \textit{Raman shift (relative wavenumber)}
• invcm2nm: conversion \textit{inverse cm (absolute wavenumber)} \rightarrow \textit{nanometers}
• invcm2ev: conversion \textit{inverse cm (absolute wavenumber)} \rightarrow \textit{electronvolt}
• invcm2freq: conversion \textit{inverse cm (absolute wavenumber)} \rightarrow \textit{frequency in THz}
• raman2invcm: conversion \textit{Raman shift (relative wavenumber)} \rightarrow \textit{inverse cm (absolute wavenumber)}
• raman2nm: conversion \textit{Raman shift (relative wavenumber)} \rightarrow \textit{nanometers}
• raman2ev: conversion \textit{Raman shift (relative wavenumber)} \rightarrow \textit{electronvolt}
• raman2freq: conversion \textit{Raman shift (relative wavenumber)} \rightarrow \textit{frequency in THz}
• ev2raman: conversion \textit{electronvolt} \rightarrow \textit{Raman shift (relative wavenumber)}
• ev2invcm: conversion \textit{electronvolt} \rightarrow \textit{inverse cm (absolute wavenumber)}
• ev2nm: conversion \textit{electronvolt} \rightarrow \textit{nanometers}
• ev2freq: conversion \textit{electronvolt} \rightarrow \textit{frequency in THz}
• freq2nm: conversion \textit{frequency in THz} \rightarrow \textit{nanometers}
• freq2invcm: conversion \textit{frequency in THz} \rightarrow \textit{inverse cm (absolute wavenumber)}
• freq2ev: conversion \textit{frequency in THz} \rightarrow \textit{electronvolt}
• freq2raman: conversion \textit{frequency in THz} \rightarrow \textit{Raman shift (relative wavenumber)}

\textbf{Author(s)}

R. Kiselev

\textbf{Examples}

\texttt{wlconv (3200, \"Raman shift\", \"nm\", laser = 785.04)}
\texttt{wlconv( 785, \"nm\", \"invcm\")}

\textbf{Description}

These Methods allow to extract and replace parts of the hyperSpec object.
Usage

```r
## S4 method for signature 'hyperSpec'
x[i, j, l, ..., wl.index = FALSE, drop = FALSE]

## S4 method for signature 'hyperSpec'
x[[i, j, l, ..., wl.index = FALSE, drop = FALSE]]

## S4 method for signature 'hyperSpec'
x$name

## S4 replacement method for signature 'hyperSpec'
x[i, j, ...] <- value

## S4 replacement method for signature 'hyperSpec'
x[[i, j, l, wl.index = FALSE, ...]] <- value

## S4 replacement method for signature 'hyperSpec'
x$name <- value
```

Arguments

- `x`: a `hyperSpec` Object
- `i`: row index: selects spectra
  - `[[` and `code`[<] accept indexing with logical matrix or a n by 2 integer index matrix. In this case the indexing is done inside the spectra matrix. See the examples below.
- `j`: selecting columns of `@data`
- `l`: selecting columns of the spectra matrix. If `l` is numeric, the default behaviour is treating `l` as wavelengths, *not* as indices.
- `...`: ignored
- `wl.index`: If `TRUE` (default), the value(s) in `l` are treated as column indices for the spectral matrix. Otherwise, the numbers in `l` are treated as wavelengths and the corresponding column indices are looked up first via `wlRi`.
- `drop`: For `[[`: drop unnecessary dimensions, see `drop` and `Extract`. Ignored for `[]`, as otherwise invalid `hyperSpec` objects might result.
- `name`: name of the data column to extract. `$spc` yields the spectra matrix.
- `value`: the replacement value

Details

They work with respect to the spectra (rows of `x`), the columns of the data matrix, and the wavelengths (columns of the spectra matrix).

Thus, they can be used for selecting/deleting spectra, cutting the spectral range, and extracting or setting the data belonging to the spectra.

Convenient shortcuts for access of the spectra matrix and the `data.frame` in slot `data` are provided.
Extracting: [, [ ], and $.

The version with single square brackets ([) returns the resulting hyperSpec object.
[[ yields data.frame of slot @data of that corresponding hyperSpec object returned with the same arguments by [ if columns were selected (i.e. j is given), otherwise the spectra matrix x@data$spc.
$ returns the selected column of the data.frame in slot @data.

Shortcuts. Three shortcuts to conveniently extract much needed parts of the object are defined:
x[[ ]] returns the spectra matrix.
x$. returns the complete slot @data, including the spectra matrix in column $spc, as a data.frame.
x$. . returns a data.frame like x$, but without the spectra matrix.

Replacing: [<=, [[<=, and $<=.

## S4 method for signature 'hyperSpec':
x [i, j, l, ...] <- value

## S4 method for signature 'hyperSpec':
x [[i, j, l, w1.index = FALSE, ...]] <- value

## S4 method for signature 'hyperSpec':
x$name <- value

value gives the values to be assigned.

For $, this can also be a list of the form list (value = value, label = label), with label containing the label for data column name.

[<= replaces parts of the spectra matrix.
[[<= replaces parts of the data.frame in slot x@data.

$<= replaces a column of the data.frame in slot x@data. The value may be a list with two elements, value and label. In this case the label of the data column is changed accordingly.

$. . <= is again an abbreviation for the data.frame without the spectra matrix.

Value

For [, [<=, [[<=, and $<= a hyperSpec object,
for [[ a matrix or data.frame, and
for $ the column of the data.frame @data.
x[[ ]] returns the complete spectra matrix.
x$. returns the complete slot @data,
x$. . returns the data.frame in @data but without the column @data$spc containing the spectra matrix.

See Also

wl2i on conversion of wavelength ranges to indices.
drop and Extract on drop.
Examples

```r
## index into the rows (spectra) ----------------------------------------
## make some "spectra"

## numeric index
plot(flu, "spc", lines.args = list(lty = 2))
plot(flu[1:3], "spc", add = TRUE, col = "red")  # select spectra
plot(flu[-(1:3)], "spc", add = TRUE, col = "blue")  # delete spectra

## logic index
index <- rnorm(6) > 0
plot(flu[index], "spc", add = TRUE, col = "red")  # select spectra
plot(flu[!index], "spc", add = TRUE, col = "blue")  # select spectra

## index into the data columns ----------------------------------------
range(chondro[,]"x"))
colnames(chondro[,]1))
dim(chondro[,]c(TRUE, FALSE, FALSE)])

## the shortcut functions -------------------------------------------

## extract the spectra matrix
flu[]

## indexing via logical matrix
summary(flu[[flu < 125]])

## indexing the spectra matrix with index matrix n by 2
ind <- matrix(c(1, 2, 4, 406, 405.5, 409), ncol = 2)
ind
flu [[ind]]

ind <- matrix(c(1, 2, 4, 4:6), ncol = 2)
ind
flu [[ind, wl.index = TRUE]]
pca <- prcomp(flu[])

## result is data.frame, if j is given:
result <- flu [[1:2, 405 ~ 410]]
result
class(result)
colnames(result)

## extract the data.frame including the spectra matrix
flu$_$
dim(flu$_$)
```
colnames (flu$.)
flu$.$spc

calibration <- lm (spc ~ c, data = flu[,450]$.)
calibration

flu$..
colnames (flu$..)

## replacement functions
spc <- flu
spc$
spc[., "c"] <- 16:11
## be careful:
plot (spc)
spc [] <- 6:1
spc$
plot (spc)

spc <- flu [, , 405 - 410]
spc [][]
spc [[3]] <- -spc[[3]]
spc [][]
spc [,,405 : 410] <- -spc[,,405 : 410]
spc [][]
spc [,,405 - 410] <- -spc[,,405 - 410]

## indexing with logical matrix
spc <- flu [, , min - 410]
spc < 125
spc [[spc < 125]] <- NA
spc [][]

## indexing with n by 2 matrix
ind <- matrix (c (1, 2, 4, 406, 405.5, 409), ncol = 2)
ind
spc [[ind]] <- 3
spc [][]

ind <- matrix (c (1, 2, 4, 4:6), ncol = 2)
ind
spc [[ind, wl.index = TRUE]] <- 9999
spc [][]

spc$
spc$z <- 1:6
spc
spc$z <- list (1:6, "z / a.u.")
Comparison, 21
decomposition, 24
dimnames, hyperSpec-method, 25
initialize, 31
Math2, hyperSpec-method, 41
ncol, hyperSpec-method, 46
plot-methods, 51
sample, hyperSpec-method, 83
scale, hyperSpec-method, 84
split, 99
sweep, 102

* Topic misc
  hy.getOptions, 28

* Topic multivar
  mean_sd, numeric-method, 43

* Topic package
  hyperSpec-package, 4

* Topic print
  as.character, hyperSpec-method, 11

* Topic programming
  hy.unittest, 29

* Topic univar
  mean_sd, numeric-method, 43

* Topic utilities
  .DollarNames.hyperSpec, 5
  hy.unittest, 29

*, hyperSpec, hyperSpec-method (Arith), 9
+, hyperSpec, hyperSpec-method (Arith), 9
-, hyperSpec, hyperSpec-method (Arith), 9
.DollarNames, 5
.DollarNames (.DollarNames.hyperSpec), 5
.DollarNames.hyperSpec-method (.DollarNames.hyperSpec), 5
/D, hyperSpec, hyperSpec-method (Arith), 9
<=, hyperSpec, hyperSpec-method (Comparison), 21
<-, hyperSpec, hyperSpec-method (Comparison), 21
==, hyperSpec, hyperSpec-method (Comparison), 21
>, hyperSpec, hyperSpec-method (Comparison), 21
>=, hyperSpec, hyperSpec-method (Comparison), 21
[. , 13, 107
[([, hyperSpec-method), 112
[, hyperSpec-method, 112

<= ([, hyperSpec-method), 112
<-, hyperSpec-method
  ([, hyperSpec-method), 112
[[ ([, hyperSpec-method), 112
[, hyperSpec-method
  ([, hyperSpec-method), 112
[[< ([, hyperSpec-method), 112
[[<-, hyperSpec-method
  ([, hyperSpec-method), 112
$(,[, hyperSpec-method), 112
$, hyperSpec-method
  ([, hyperSpec-method), 112
$<-[,[, hyperSpec-method), 112
$<-, hyperSpec-method
  ([, hyperSpec-method), 112
%% (Arith), 9
%%, hyperSpec, hyperSpec-method (Arith), 9
%%, hyperSpec, matrix-method (Arith), 9
%%, matrix, hyperSpec-method (Arith), 9
%%, hyperSpec, hyperSpec-method (Arith), 9
%%, hyperSpec, hyperSpec-method (Arith), 9
%%, 25
^, hyperSpec, hyperSpec-method (Arith), 9
abline, 60
abs, hyperSpec-method
  (Math2, hyperSpec-method), 41
acos, hyperSpec-method
  (Math2, hyperSpec-method), 41
acosh, hyperSpec-method
  (Math2, hyperSpec-method), 41
aggregate, 5, 6, 8
aggregate, hyperSpec-method (aggregate), 5
all, hyperSpec-method (Summary), 101
all.equal, 21, 22, 28, 102
all.equal (Comparison), 21
all.equal, hyperSpec, hyperSpec-method (Comparison), 21
alois.palette (matlab.palette), 42
any, hyperSpec-method (Summary), 101
apply, 7, 7, 8
apply, hyperSpec-method (apply), 7
Arith, 9, 22, 42
Arith, hyperSpec, hyperSpec-method (Arith), 9
Arith, hyperSpec, matrix-method (Arith), 9
INDEX

Arith, hyperSpec, missing-method (Arith), 9
Arith, hyperSpec, numeric-method (Arith), 9
Arith, hyperSpec-method (Arith), 9
Arith, matrix, hyperSpec-method (Arith), 9
Arith, numeric, hyperSpec-method (Arith), 9
Arithmetic, 10
as.character, 12
as.character
  (as.character, hyperSpec-method), 11
as.character, hyperSpec-method, 11
as.data.frame, 10, 12, 13
as.data.frame, hyperSpec-method
  (as.data.frame), 12
as.data.frame, hyperSpec
  (as.data.frame), 12
as.dist, 51
as.long.df (as.data.frame), 12
as.matrix, 13, 84
as.matrix (as.data.frame), 12
as.matrix, hyperSpec-method
  (as.data.frame), 12
as.matrix, hyperSpec (as.data.frame), 12
as.t.df (as.data.frame), 12
as.wide.df (as.data.frame), 12
asin, hyperSpec-method
  (Math2, hyperSpec-method), 41
asinh, hyperSpec-method
  (Math2, hyperSpec-method), 41
atan, hyperSpec-method
  (Math2, hyperSpec-method), 41
atanh, hyperSpec-method
  (Math2, hyperSpec-method), 41
ave, 6
ave, hyperSpec-method (aggregate), 5
axis, 59, 60
axis.break, 59, 60

barbiturates, 14
bind, 15, 99
cBind, 15, 16
cbind, 16, 45, 67
cbind, hyperSpec (bind), 15
cbind2, 16
cbind2, hyperSpec, hyperSpec-method
  (bind), 15
cbind2, hyperSpec, missing-method (bind), 15
ceiling, hyperSpec-method
  (Math2, hyperSpec-method), 41
chk.hy, 17
chondro, 18, 55
collapse, 16, 19, 28, 45
colMeans, 20
colMeans, hyperSpec-method (colSums), 20
colmix.rgb, 35
colmix.rgb (legendright), 34
colnames, 26
colnames (dimnames, hyperSpec-method), 25
colnames, hyperSpec-method
  (dimnames, hyperSpec-method), 25
colnames<- (dimnames, hyperSpec-method), 25
colnames<-, hyperSpec-method
  (dimnames, hyperSpec-method), 25
colorRampPalette, 42
colSums, 20, 20
colSums, hyperSpec-method (colSums), 20
Compare, hyperSpec, hyperSpec-method
  (Comparison), 21
Compare, hyperSpec, matrix-method
  (Comparison), 21
Compare, hyperSpec, numeric-method
  (Comparison), 21
Compare, hyperSpec-method (Comparison), 21
Compare, matrix, hyperSpec-method
  (Comparison), 21
Compare, numeric, hyperSpec-method
  (Comparison), 21
Comparison, 10, 21, 22, 42
connections, 79
contour, 57
cos, hyperSpec-method
  (Math2, hyperSpec-method), 41
cosh, hyperSpec-method
  (Math2, hyperSpec-method), 41
cov, 23, 82
cov, hyperSpec, missing-method, 23
create (initialize), 31
create, hyperSpec-method (initialize), 31
cumsum, hyperSpec-method
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>cummin, hyperSpec-method</td>
<td>(Math2, hyperSpec-method)</td>
<td>41</td>
</tr>
<tr>
<td>cumprod, hyperSpec-method</td>
<td>(Math2, hyperSpec-method)</td>
<td>41</td>
</tr>
<tr>
<td>cumsum, hyperSpec-method</td>
<td>(Math2, hyperSpec-method)</td>
<td>41</td>
</tr>
<tr>
<td>decomposition</td>
<td></td>
<td>24, 105</td>
</tr>
<tr>
<td>digamma, hyperSpec-method</td>
<td>(Math2, hyperSpec-method)</td>
<td>41</td>
</tr>
<tr>
<td>dim, hyperSpec-method</td>
<td></td>
<td>47</td>
</tr>
<tr>
<td>dimnames</td>
<td>(ncol, hyperSpec-method)</td>
<td>46</td>
</tr>
<tr>
<td>dimnames</td>
<td>(dimnames, hyperSpec-method)</td>
<td>25</td>
</tr>
<tr>
<td>dimnames</td>
<td>(dimnames, hyperSpec-method, hyperSpec-method)</td>
<td>25</td>
</tr>
<tr>
<td>drop</td>
<td></td>
<td>83, 113, 114</td>
</tr>
<tr>
<td>empty</td>
<td></td>
<td>27</td>
</tr>
<tr>
<td>exp, hyperSpec-method</td>
<td>(Math2, hyperSpec-method)</td>
<td>41</td>
</tr>
<tr>
<td>expm1, hyperSpec-method</td>
<td>(Math2, hyperSpec-method)</td>
<td>41</td>
</tr>
<tr>
<td>export (read.txt.long)</td>
<td></td>
<td>78</td>
</tr>
<tr>
<td>fitraster (makeraster)</td>
<td></td>
<td>36</td>
</tr>
<tr>
<td>floor, hyperSpec-method</td>
<td>(Math2, hyperSpec-method)</td>
<td>41</td>
</tr>
<tr>
<td>flu</td>
<td></td>
<td>27</td>
</tr>
<tr>
<td>freq2ev (wlconv)</td>
<td></td>
<td>110</td>
</tr>
<tr>
<td>freq2invcm (wlconv)</td>
<td></td>
<td>110</td>
</tr>
<tr>
<td>freq2nm (wlconv)</td>
<td></td>
<td>110</td>
</tr>
<tr>
<td>freq2raman (wlconv)</td>
<td></td>
<td>110</td>
</tr>
<tr>
<td>gamma, hyperSpec-method</td>
<td>(Math2, hyperSpec-method)</td>
<td>41</td>
</tr>
<tr>
<td>geom_line, geom_point, geom_tile</td>
<td></td>
<td>65, 34, 35, 62, 35, 63, 34, 35, 63, 65, 38, 37, 38, 38</td>
</tr>
</tbody>
</table>
log, hyperSpec-method
  (Math2, hyperSpec-method), 41
log1p, hyperSpec-method
  (Math2, hyperSpec-method), 41
make.names, 66
makeraster, 36
map.identify, 28, 38, 95
map.identify(plotmap), 54
map.sel.poly, 37, 56, 95
mark.dendrogram, 39
markpeak, 40
Math, 8, 10, 22, 42
Math (Math2, hyperSpec-method), 41
Math, hyperSpec-method
  (Math2, hyperSpec-method), 41
Math2 (Math2, hyperSpec-method), 41
Math2, hyperSpec-method, 41
matlab.dark.palette (matlab.palette), 42
matlab.palette, 42
matmult, 10
matplot, 52
max, hyperSpec-method (Summary), 101
mean, 44
mean, hyperSpec-method
  (mean_sd, numeric-method), 43
mean_pm_sd (mean_sd, numeric-method), 43
mean_pm_sd, hyperSpec-method
  (mean_sd, numeric-method), 43
mean_pm_sd, matrix-method
  (mean_sd, numeric-method), 43
mean_pm_sd, numeric-method
  (mean_sd, numeric-method), 43
mean_sd (mean_sd, numeric-method), 43
mean_sd, hyperSpec-method
  (mean_sd, numeric-method), 43
mean_sd, matrix-method
  (mean_sd, numeric-method), 43
mean_sd, numeric-method, 43
melt, 13
merge, 16, 19, 45
merge
  (merge, hyperSpec, hyperSpec-method), 45
merge, hyperSpec, hyperSpec-method, 45
merge.data.frame, 45
min, hyperSpec-method (Summary), 101
ncol, 47
ncol, hyperSpec-method, 46
new, 31
new(initialize), 31
new, hyperSpec-method(initialize), 31
nm2ev (wlconv), 110
nm2freq (wlconv), 110
nm2invcm (wlconv), 110
nm2raman (wlconv), 110
normalize.colrange (legendright), 34
normalize.minmax (legendright), 34
normalize.null (legendright), 34
normalize.range (legendright), 34
normalize01, 28, 47, 103, 108
normalize01, hyperSpec-method
  (normalize01), 47
normalize01, matrix-method
  (normalize01), 47
normalize01, numeric-method
  (normalize01), 47
nrow, 47
nrow, hyperSpec-method
  (ncol, hyperSpec-method), 46
nwl (ncol, hyperSpec-method), 46
Options (Comparison), 21
order, 49, 78
orderwl, 45, 48, 109
panel.identify, 56
panel.levelplot.raster, 55
panel.voronoi, 55, 56
par, 59, 60, 94
paracetamol, 49
pearson.dist, 50
plot, 52, 56, 59, 60
plot (plot-methods), 51
plot, ANY, ANY-method (plot-methods), 51
plot, hyperSpec, character-method
  (plot-methods), 51
plot, hyperSpec, missing-method
  (plot-methods), 51
plot-methods, 51
plotc, 25, 51, 52, 52, 62
plotmap, 25, 38, 51, 52, 54, 63
plotmat, 52, 57
plotmath, 31, 33
plotspc, 25, 51, 52, 58, 58, 60, 65, 93, 95
plotvoronoi, 38, 52
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>plotvoronoi (plotmap)</td>
<td>54</td>
</tr>
<tr>
<td>point.in.polygon</td>
<td>38</td>
</tr>
<tr>
<td>polygon</td>
<td>59, 60</td>
</tr>
<tr>
<td>pooled.cov</td>
<td>82</td>
</tr>
<tr>
<td>pooled.cov (cov,hyperSpec,missing-method)</td>
<td>23</td>
</tr>
<tr>
<td>prcomp</td>
<td>25</td>
</tr>
<tr>
<td>princomp</td>
<td>25</td>
</tr>
<tr>
<td>print</td>
<td>12</td>
</tr>
<tr>
<td>print (as.character,hyperSpec-method)</td>
<td>11</td>
</tr>
<tr>
<td>print,hyperSpec-method</td>
<td>11</td>
</tr>
<tr>
<td>prod,hyperSpec-method (Summary)</td>
<td>101</td>
</tr>
<tr>
<td>qmixlegend</td>
<td>64</td>
</tr>
<tr>
<td>qmixlegend (legendright)</td>
<td>34</td>
</tr>
<tr>
<td>qmixtile</td>
<td>64</td>
</tr>
<tr>
<td>qmixtile (legendright)</td>
<td>34</td>
</tr>
<tr>
<td>qplotc</td>
<td>61</td>
</tr>
<tr>
<td>qplotmap</td>
<td>63</td>
</tr>
<tr>
<td>qplotmixmap</td>
<td>64</td>
</tr>
<tr>
<td>qplotspec</td>
<td>65</td>
</tr>
<tr>
<td>quantile</td>
<td>44</td>
</tr>
<tr>
<td>quantile,hyperSpec-method</td>
<td>43</td>
</tr>
<tr>
<td>quantile,hyperSpec-method (mean_sd,numeric-method)</td>
<td>43</td>
</tr>
<tr>
<td>R.matlab</td>
<td>79, 80</td>
</tr>
<tr>
<td>rainbow</td>
<td>42, 43</td>
</tr>
<tr>
<td>raman2ev (wlconv)</td>
<td>110</td>
</tr>
<tr>
<td>raman2freq (wlconv)</td>
<td>110</td>
</tr>
<tr>
<td>raman2invcm (wlconv)</td>
<td>110</td>
</tr>
<tr>
<td>raman2nm (wlconv)</td>
<td>110</td>
</tr>
<tr>
<td>range,hyperSpec-method (Summary)</td>
<td>101</td>
</tr>
<tr>
<td>rBind</td>
<td>15, 16</td>
</tr>
<tr>
<td>rbind</td>
<td>16, 19, 28, 45, 66, 67</td>
</tr>
<tr>
<td>rbind.fill</td>
<td>19, 67</td>
</tr>
<tr>
<td>rbind.fill (rbind.fill.matrix)</td>
<td>66</td>
</tr>
<tr>
<td>rbind.fill.matrix</td>
<td>66</td>
</tr>
<tr>
<td>rbind.hyperSpec (bind)</td>
<td>15</td>
</tr>
<tr>
<td>rbind2</td>
<td>16</td>
</tr>
<tr>
<td>rbind2,hyperSpec,hyperSpec-method (bind)</td>
<td>15</td>
</tr>
<tr>
<td>rbind2,hyperSpec,missing-method (bind)</td>
<td>15</td>
</tr>
<tr>
<td>read.cytomat (read.mat.Cytospec)</td>
<td>72</td>
</tr>
<tr>
<td>read.ENVI</td>
<td>28, 69, 70, 79, 80</td>
</tr>
<tr>
<td>read.ENVI.Nicolet</td>
<td>67</td>
</tr>
<tr>
<td>read.ini</td>
<td>70</td>
</tr>
<tr>
<td>read.jdx</td>
<td>71</td>
</tr>
<tr>
<td>read.mat.Cytospec</td>
<td>72</td>
</tr>
<tr>
<td>read.spc</td>
<td>73, 75, 79, 80</td>
</tr>
<tr>
<td>read.spc.Kaiser</td>
<td>74</td>
</tr>
<tr>
<td>read.spc.KaiserLowHigh</td>
<td>(read.spc.Kaiser)</td>
</tr>
<tr>
<td>read.spc.KaiserMap</td>
<td>(read.spc.Kaiser)</td>
</tr>
<tr>
<td>read.spe</td>
<td>75</td>
</tr>
<tr>
<td>read.table</td>
<td>78–80</td>
</tr>
<tr>
<td>read.txt.Horiba</td>
<td>77</td>
</tr>
<tr>
<td>read.txt.long</td>
<td>78, 87</td>
</tr>
<tr>
<td>read.txt.Shimadzu</td>
<td>81</td>
</tr>
<tr>
<td>read.txt.wide</td>
<td>77, 87</td>
</tr>
<tr>
<td>read.txt.wide (read.txt.long)</td>
<td>78</td>
</tr>
<tr>
<td>readLines</td>
<td>71, 81</td>
</tr>
<tr>
<td>readMat</td>
<td>79</td>
</tr>
<tr>
<td>rect</td>
<td>39</td>
</tr>
<tr>
<td>rmvnorm</td>
<td>81</td>
</tr>
<tr>
<td>rmvnorm,hyperSpec-method</td>
<td>(rmvnorm)</td>
</tr>
<tr>
<td>rmvnorm,numeric,hyperSpec,array-method</td>
<td>(rmvnorm)</td>
</tr>
<tr>
<td>rmvnorm,numeric,hyperSpec,matrix-method</td>
<td>(rmvnorm)</td>
</tr>
<tr>
<td>rmvnorm,numeric,matrix,array-method</td>
<td>(rmvnorm)</td>
</tr>
<tr>
<td>rmvnorm,numeric,matrix,matrix-method</td>
<td>(rmvnorm)</td>
</tr>
<tr>
<td>round</td>
<td>81, 82</td>
</tr>
<tr>
<td>round,hyperSpec-method</td>
<td>(Math2,hyperSpec-method)</td>
</tr>
<tr>
<td>rowMeans</td>
<td>20</td>
</tr>
<tr>
<td>rowMeans,hyperSpec-method</td>
<td>(colSums)</td>
</tr>
<tr>
<td>rownames</td>
<td>26</td>
</tr>
<tr>
<td>rownames (dimnames,hyperSpec-method)</td>
<td>25</td>
</tr>
<tr>
<td>rownames,hyperSpec-method</td>
<td>(dimnames,hyperSpec-method)</td>
</tr>
<tr>
<td>rownames&lt;- (dimnames,hyperSpec-method)</td>
<td>25</td>
</tr>
<tr>
<td>rownames&lt;-,hyperSpec-method</td>
<td>(dimnames,hyperSpec-method)</td>
</tr>
<tr>
<td>rowSums</td>
<td>20</td>
</tr>
<tr>
<td>rowSums,hyperSpec-method</td>
<td>(colSums)</td>
</tr>
<tr>
<td>S4groupGeneric</td>
<td>10, 22, 42</td>
</tr>
<tr>
<td>sample</td>
<td>84</td>
</tr>
</tbody>
</table>
INDEX

sample, data.frame-method
  (sample, hyperSpec-method), 83
sample, hyperSpec-method, 83
sample, matrix-method
  (sample, hyperSpec-method), 83
save, 78
scale, 85
scale (scale, hyperSpec-method), 84
scale, hyperSpec-method, 84
scale-methods (scale, hyperSpec-method), 84
scan, 86–88
scan.asc.Andor, 85
scan.dat.Witec (scan.txt.Witec), 88
scan.txt.Renishaw, 80, 86
scan.txt.Witec, 88
scan.zip.Renishaw (scan.txt.Renishaw), 86
sd, 44
sel.poly, 38
sel.poly (map.sel.poly), 37
seq, 89, 90
seq (seq.hyperSpec), 89
seq, hyperSpec-method (seq.hyperSpec), 89
seq.hyperSpec, 89
seq.int, 89
seq_along, 89
show, 12
show (as.character, hyperSpec-method), 11
show, hyperSpec-method
  (as.character, hyperSpec-method), 11
sign, hyperSpec-method
  (Math2, hyperSpec-method), 41
signif, 106, 107
signif, hyperSpec-method
  (Math2, hyperSpec-method), 41
sin, hyperSpec-method
  (Math2, hyperSpec-method), 41
sinh, hyperSpec-method
  (Math2, hyperSpec-method), 41
smooth spline, 97–99
spc.bin, 90, 107
spc.fit.poly, 91, 98
spc.fit.poly.below, 98
spc.identify, 28, 56, 60, 93
spc.label.default (spc.identify), 93
spc.label.wlonly (spc.identify), 93
spc.loess, 96, 99, 107
spc.NA.linapprox, 97
spc.point.default (spc.identify), 93
spc.point.max (spc.identify), 93
spc.point.min (spc.identify), 93
spc.point.sqr (spc.identify), 93
spc.rubberband, 97
spc.smooth.spline, 98
spc.showcalpoints (read.spe), 75
split, 99, 100
split, ANY-method (split), 99
split, hyperSpec-method (split), 99
split-methods (split), 99
sqrt, hyperSpec-method
  (Math2, hyperSpec-method), 41
stack, 13
stacked.offsets, 59
stacked.offsets (plotspec), 58
subset, 100, 100, 101
subset, hyperSpec-method (subset), 100
sum, hyperSpec-method (Summary), 101
Summary, 42, 101, 102
summary, 12
summary
  (as.character, hyperSpec-method), 11
Summary, hyperSpec-method (Summary), 101
summary, hyperSpec-method
  (as.character, hyperSpec-method), 11
svUnit, 29
sweep, 10, 22, 102, 102, 103
sweep, hyperSpec-method (sweep), 102
sweep-methods (sweep), 102
tan, hyperSpec-method
  (Math2, hyperSpec-method), 41
tanh, hyperSpec-method
  (Math2, hyperSpec-method), 41
tapply, 6
text, 39, 93, 94
textio, 70, 74
title, 59, 60
trellis.factor.key, 55, 56, 103
trigamma, hyperSpec-method
  (Math2, hyperSpec-method), 41
trunc, hyperSpec-method
  (Math2, hyperSpec-method), 41
unit, 55
unzip, 79

validObject, 17
validObject (chk.hy), 17
validObject, hyperSpec-method (chk.hy), 17
vanderMonde, 48, 104, 108
vanderMonde, hyperSpec-method (vanderMonde), 104

wc, 106
wl, 25, 26, 106
wl.eval, 48, 105, 108
wl2i, 89, 90, 109, 113, 114
wl<-(wl), 106
wlconv, 110
write.table, 78–80
write.txt.long (read.txt.long), 78
write.txt.wide (read.txt.long), 78
writeMat, 79

xyplot, 53