

# Package ‘hyperSpec’

May 8, 2026

**Encoding** UTF-8

**Type** Package

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

**Version** 0.100.3

**Date** 2025-10-12

**Maintainer** Claudia Beleites <Claudia.Beleites@chemometrix.gmbh>

**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 (>= 3.6.0), lattice, grid, ggplot2 (>= 2.2.0), xml2

**Suggests** R.matlab, deldir, rgl, plotrix, sp, baseline, compiler, inline, Repp, MASS, fastcluster, pls, mvtnorm, digest, reshape, devtools, R.rsp, tibble, knitr, rmarkdown

**Imports** testthat, methods, utils, latticeExtra, lazyeval, dplyr, rlang

**URL** <https://r-hyperspec.github.io/hyperSpec/> (documentation),  
<https://github.com/r-hyperspec/hyperSpec> (code)

**BugReports** <https://github.com/r-hyperspec/hyperSpec/issues>

**VignetteBuilder** knitr

**Collate** 'validate.R' 'hyperspec-class.R' 'unittest.R' 'paste.row.R'  
'Arith.R' 'Compare.R' 'DollarNames.R' 'Math.R' 'chk.hy.R'  
'read.txt.wide.R' 'read.txt.long.R' 'options.R' 'wl.R'  
'fileio.optional.R' 'NEW-functions.R' 'Summary.R' 'aggregate.R'

'all.equal.R' 'apply.R' 'as.data.frame.R' 'barbiturates.R'  
 'bind.R' 'call.list.R' 'chondro.R' 'colMeans.R' 'collapse.R'  
 'count\_lines.R' 'cov.R' 'decomposition.R' 'deprecated.R'  
 'dim.R' 'dimnames.R' 'droplevels.R' 'empty.R' 'wl2i.R'  
 'extract.R' 'factor2num.R' 'fix\_spc\_colnames.R' 'flu.R'  
 'getbynames.R' 'regexps.R' 'guesswavelength.R' 'paracetamol.R'  
 'laser.R' 'hyperspec-package.R' 'initialize.R' 'labels.R'  
 'plotmap.R' 'levelplot.R' 'makeraster.R' 'map.identify.R'  
 'map.sel.poly.R' 'mark.dendrogram.R' 'mark.peak.R'  
 'matlab.palette.R' 'mean\_sd.R' 'merge.R' 'mvtnorm.R'  
 'normalize01.R' 'orderwl.R' 'pearson.dist.R' 'plot.R' 'plotc.R'  
 'plotmat.R' 'plotspc.R' 'plotvoronoi.R' 'qplot.R'  
 'qplotmixmap.R' 'quantile.R' 'rbind.fill.R' 'read.ENVI.R'  
 'read.ENVI.HySpex.R' 'read.ENVI.Nicolet.R' 'read.txt.Witec.R'  
 'read.asc.Andor.R' 'read.asc.PerkinElmer.R' 'read.ini.R'  
 'read.jdx.R' 'read.mat.Cytospec.R' 'read.mat.Witec.R'  
 'read.spc.Kaiser.R' 'read.spc.R' 'read.spc.Shimadzu.R'  
 'read.spe.R' 'read.txt.Horiba.R' 'read.txt.Renishaw.R'  
 'read.txt.Shimadzu.R' 'replace.R' 'sample.R' 'scale.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'  
 'vandermonde.R' 'wc.R' 'wleval.R' 'write.txt.long.R'  
 'write.txt.wide.R' 'y-pastenames.R' 'zzz.R'

### **RoxygenNote** 7.2.3

**NeedsCompilation** no

**Author** Claudia Beleites [aut, cre, dtc] (ORCID:  
<https://orcid.org/0000-0003-1626-154X>),  
 Valter Sergio [aut],  
 Alois Bonifacio [ctb, dtc],  
 Marcel Dahms [ctb],  
 Björn Egert [ctb],  
 Simon Fuller [ctb],  
 Vilmantas Gegzna [aut],  
 Rustam Guliev [ctb],  
 Bryan A. Hanson [ctb],  
 Michael Hermes [ctb],  
 Martin Kammer [dtc],  
 Roman Kiselev [ctb],  
 Sebastian Mellor [ctb]

**Repository** CRAN

**Date/Publication** 2025-10-13 05:10:33 UTC

## Contents

hyperSpec-package	5
.cluster.wavelengths	5
.collapse.equal	6
.DollarNames.hyperSpec	6
.fix_spc_colnames	7
.read.spe.xml	7
.read.spe.xml_string	8
aggregate	8
apply	10
Arith	12
as.character,hyperSpec-method	14
as.data.frame	15
as.hyperSpec	17
barbiturates	18
bind	19
chk.hy	21
chondro	22
collapse	23
colSums	24
Comparison	25
count_lines	27
cov,hyperSpec,missing-method	28
decomposition	29
dimnames,hyperSpec-method	31
droplevels,hyperSpec-method	32
empty	33
flu	33
Future-functions	34
guess.wavelength	35
hy.getOptions	35
hy.unittest	37
hyperSpec-class	37
initialize	38
labels<-	40
laser	41
legendright	42
makeraster	44
map.sel.poly	45
mark.dendrogram	47
markpeak	48
Math2,hyperSpec-method	49
matlab.palette	50
mean_sd,numeric-method	51
merge,hyperSpec,hyperSpec-method	53
ncol,hyperSpec-method	54
normalize01	56

orderwl	57
paracetamol	58
pearson.dist	58
plot-methods	59
plotc	61
plotmap	62
plotmat	65
plotspc	67
qplotc	71
qplotmap	72
qplotmixmap	73
qplotspc	74
rbind.fill.matrix	75
read.asc.PerkinElmer	76
read.cytomat	77
read.ENVI	78
read.ini	81
read.jdx	82
read.spc	83
read.spc.Kaiser	85
read.spe	86
read.txt.Horiba	87
read.txt.Shimadzu	88
read.txt.wide	89
rmmvnorm	93
sample,hyperSpec-method	94
scale,hyperSpec-method	95
seq.hyperSpec	96
spc.bin	97
spc.fit.poly	99
spc.identify	100
spc.loess	104
spc.NA.approx	105
spc.rubberband	106
spc.smooth.spline	107
split	108
subset	109
Summary	110
sweep	111
trellis.factor.key	112
vanderMonde	113
wc	114
wl	115
wl.eval	116
wl2i	117
wlconv	119
[,hyperSpec-method	121

---

hyperSpec-package      *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.

---

`.cluster.wavelengths`      *Find clusters of approximately equal wavelengths*

---

### Description

Find clusters of approximately equal wavelengths

### Usage

```
.cluster.wavelengths(dots, wl.tolerance)
```

### Arguments

`dots`                      list of hyperSpec objects to collapse

`wl.tolerance`              wavelength difference tolerance

**Value**

data.frame with information about suitable wavelength bins

---

<code>.collapse.equal</code>	<i>Try finding groups of hyperSpec objects with (approximately) equal wavelength axes</i>
------------------------------	---

---

**Description**

... and directly `rbind.fill` them.

**Usage**

```
.collapse.equal(dots, wl.tolerance)
```

**Arguments**

<code>dots</code>	list with hyperSpec object to collapse
<code>wl.tolerance</code>	wavelength difference tolerance

**Value**

possible shorter list of dots

---

<code>.DollarNames.hyperSpec</code>	<i>command line completion for \$</i>
-------------------------------------	---------------------------------------

---

**Description**

command line completion for \$

**Usage**

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

**Arguments**

<code>x</code>	the hyperSpecobject
<code>pattern</code>	pattern to look for

**Value**

the name of the extra data slot

**Author(s)**

C. Beleites

**See Also**

[.DollarNames](#)

---

.fix\_spc\_colnames      *Ensure that the spectra matrix has the wavelengths in column names*

---

**Description**

Ensure that the spectra matrix has the wavelengths in column names

**Usage**

```
.fix_spc_colnames(spc)
```

**Arguments**

spc                    hyperSpec object

**Value**

hyperSpec object with wavelengths in column names of \$spc

---

.read.spe.xml              *Read XML footer from SPE file format version 3.0*

---

**Description**

The new SPE file format, introduced in 2012, was designed to be backwards compatible with the previous format 2.5. The most prominent change is the new plain text XML footer holding vast experimental metadata that gets attached at the end of the file. Thus, the file contains 3 blocks: a 4100-bytes long binary header, a chunk with spectral data, and the XML footer. This function retrieves the XML footer converted to R list, and throws error if it is not available. The file format specification is available at Princeton Instruments FTP server under name 'SPE 3.0 File Format Specification'.

**Usage**

```
.read.spe.xml(filename)
```

**Arguments**

filename                - SPE filename

**Details**

This function relies on R package `xml2` to work correctly

**Value**

xml data from the file converted to R list

---

```
.read.spe.xml_string .read.spe.xml_string
```

---

**Description**

Read XML footer from SPE file format version 3.0 and return it as a long string for subsequent parsing. Basically the purpose of this function is to check that the file format version is 3.0 or above, and to find and read the correct part of this file.

**Usage**

```
.read.spe.xml_string(filename)
```

**Arguments**

`filename` - SPE filename

**Value**

string containing XML footer

---

```
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**

<code>x</code>	a hyperSpec object
<code>by</code>	grouping for the rows of <code>x@data</code> . Either a list containing an index vector for each of the subgroups or a vector that can be split in such a list.
<code>FUN</code>	function to compute the summary statistics
<code>...</code>	further arguments passed to <code>FUN</code>
<code>out.rows</code>	number of rows in the resulting hyperSpec object, for memory preallocation.
<code>append.rows</code>	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.
<code>by.isindex</code>	If a list is given in <code>by</code> : does the list already contain the row indices of the groups? If <code>FALSE</code> , the list in <code>by</code> is computed first (as in <a href="#">aggregate</a> ).

**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
```

```

color <- c("red", "blue", "black")
by <- as.factor (c (1, 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	<i>apply Computes summary statistics for the spectra of a hyperSpec object.</i>
-------	---

---

## Description

apply gives the functionality of [apply](#) for hyperSpec objects.

## Usage

```

## S4 method for signature 'hyperSpec'
apply(
  X,
  MARGIN,
  FUN,
  ...,
  label.wl = NULL,

```

```

    label.spc = NULL,
    new.wavelength = NULL,
    simplify
)

```

### Arguments

<code>X</code> , <code>spc</code>	a <code>hyperSpec</code> object
<code>MARGIN</code>	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 <code>hyperSpec</code> objects (see <a href="#">Math</a> ). If <code>MARGIN</code> is missing, the whole spectra matrix is handed to FUN, see also the examples.
<code>FUN</code>	function to compute the summary statistics
<code>...</code>	further arguments passed to FUN
<code>label.wl</code> , <code>label.spc</code>	new labels for wavelength and spectral intensity axes
<code>new.wavelength</code>	for <code>MARGIN = 2</code> : numeric vector or name of the argument in <code>...</code> that is to be used (character) as wavelength axis of the resulting object.
<code>simplify</code>	ignored: apply for <code>hyperSpec</code> results are always simplified

### 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

*Arithmetical 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**

```
flu + flu
1 / flu
all((flu + flu - 2 * flu)[[]] == 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,

**Usage**

```
## 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 than 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

**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	<i>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</i>
---------------	---

---

**Description**

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.

**Usage**

```
## 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, wl.prefix = "")

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
wl.prefix	prefix to prepend wavelength column names
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](#)

and `base::as.matrix()`

`[[()]] ([[ ]])` for a shortcut to `as.matrix`

`stack` and `melt` or `reshape2::melt()` for other functions producing long-format data.frames.

**Examples**

```

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))
}

```

---

as.hyperSpec

*as.hyperSpec: convenience conversion functions*


---

**Description**

These functions are shortcuts to convert other objects into hypeSpec objects.

**Usage**

```

as.hyperSpec(X, ...)

## S4 method for signature 'matrix'
as.hyperSpec(X, wl = guess.wavelength(colnames(X)), ...)

## S4 method for signature 'data.frame'
as.hyperSpec(
  X,
  spc = NULL,
  wl = guess.wavelength(spc),
  labels = attr(X, "labels"),
  ...
)

```

**Arguments**

<code>X</code>	the object to convert. A matrix is assumed to contain the spectra matrix, a data.frame is assumed to contain extra data.
<code>...</code>	additional parameters that should be handed over to new ("hyperSpec") (initialize)
<code>wl</code>	wavelength vector. Defaults to guessing from the column names in <code>X</code>
<code>spc</code>	spectra matrix
<code>labels</code>	list with labels

**Value**

hyperSpec object

**Note**

*Note that the behaviour of `as.hyperSpec(X)` was changed: it now assumes `X` to be extra data, and returns a hyperSpec object with 0 wavelengths. To get the old behaviour*

**See Also**

[initialize](#)

**Examples**

```
tmp <- data.frame(flu [[, , 400 ~ 410]])
(wl <- colnames (tmp))
guess.wavelength (wl)
```

---

barbiturates	<i>Barbiturates Spectra from .spc example files A time series of mass spectra in a list of hyperSpec objects.</i>
--------------	---

---

**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>

## Examples

```
barbiturates [1:3]
length (barbiturates)

barb <- collapse (barbiturates, collapse.equal = FALSE)
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 = "#00000000", 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*


---

## Description

The former difficulties with binding S4 objects are resolved since R version 3.2.0 and cbind and rbind now work as intended and expected for hyperSpec objects.

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.

## Usage

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

## S3 method for class 'hyperSpec'
cbind(...)

## S3 method for class 'hyperSpec'
rbind(...)
```

```
## S4 method for signature 'hyperSpec,hyperSpec'
cbind2(x, y)

## S4 method for signature 'hyperSpec,missing'
cbind2(x, y)

## S4 method for signature 'hyperSpec,hyperSpec'
rbind2(x, y, wl.tolerance = hy.getOption("wl.tolerance"))

## S4 method for signature 'hyperSpec,missing'
rbind2(x, y, wl.tolerance)
```

### Arguments

direction	"r" or "c" to bind rows or columns
...	The hyperSpec objects to be combined. Alternatively, <i>one</i> 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

Therefore, calling `rbind.hyperSpec` and `cbind.hyperSpec` is now deprecated: `cbind` and `rbind` should now be called directly.

However, in consequence it is no longer possible to call `cbind` or `rbind` with a list of hyperSpec objects. In that case, use `bind` or `do.call` (see example).

`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

[rbind2](#), [cbind2](#) [rbind](#), [cbind](#)

[merge](#) and [collapse](#) for combining objects that do not share spectra or wavelengths, respectively.

**Examples**

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

x <- chondro[, , 600 : 605]
x$a <- 1
x@data <- x@data[, , sample (ncol (x), ncol (x))] # reorder columns

y <- chondro [nrow (chondro) : 1, , 1730 : 1750] # reorder rows
y$b <- 2

cbind2 (x, y) # works

y$y[3] <- 5
try (cbind2 (x, y)) # error

# list of hyperSpec objects

lhy <- list (flu, flu)
do.call ("rbind", lhy)
bind ("r", lhy)
```

---

chk.hy

*Validation of hyperSpec objects*

---

**Description**

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

**Usage**

```
chk.hy(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.hy (chondro)
validObject (chondro)
```

---

chondro	<i>Raman spectra of 2 Chondrocytes in Cartilage A Raman-map (laterally resolved Raman spectra) of chondrocytes in cartilage.</i>
---------	--

---

**Description**

See the vignette vignette ("chondro", package = "hyperSpec").

**Usage**

```
chondro
```

**Format**

The data set has 875 Raman spectra measured on a  $25 \times 35$  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.  $600 - 1800 \text{ cm}^{-1}$ .

**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**

```
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

*Collapse hyperSpec objects***Description**

collapse/bind several hyperSpec objects into one object

**Usage**

```

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

```

**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.

collapse.equal logical indicating whether to try first finding groups of spectra with (approximately) equal wavelength axes. If the data is known to contain few or no such groups, collapse() will be faster with this first pass being turned off.

**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, wl.tolerance is used to determine which difference in the wavelengths is tolerated as equal: clusters of approximately equal wavelengths will span at most  $2 * wl.tolerance$ . Differences up to  $\pm wl.tolerance$  are considered equal.

The returned object has wavelengths that are the weighted average (by number of spectra) of the wavelengths within any such cluster of approximately equal wavelengths.

Labels will be taken from the first object where they are encountered. However, the order of processing objects is not necessarily the same as the order of objects in the input: collapse first processes groups of input objects that share all wavelengths (within 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. If the wavelengths are names, names are preserved and taken from the first object where they were encountered, the same applies to possible column names of the spectra matrix.

**Value**

a hyperSpec object

**Author(s)**

C. Beleites

**See Also**

[merge\(\)](#), [rbind\(\)](#), and [plyr::rbind.fill\(\)](#)

**Examples**

```
barbiturates [1:3]
collapse (barbiturates [1:3])

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

a
b
c
collapse (a, b, c)

collapse (barbiturates [1:3], collapse.equal = FALSE)
```

---

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](#)

**Examples**

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

---

Comparison

*Comparison of hyperSpec objects*

---

**Description**

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

**Usage**

```
## 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 <a href="#">all.equal</a> when testing the slots of the hyperSpec objects
check.attributes, check.names	see <a href="#">all.equal</a>
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[[ ]])
```

---

count_lines	<i>count lines (of an ASCII file)</i>
-------------	---------------------------------------

---

### Description

count lines (of an ASCII file)

### Usage

```
count_lines(file, chunksize = 10000)
```

### Arguments

file	the file name or connection
chunksize	file is read in chunks of chunksize lines.

**Value**

number of lines in file

**Author(s)**

C. Beleites

---

cov,hyperSpec,missing-method

*Covariance matrices for hyperSpec objects*

---

**Description**

Covariance matrices for hyperSpec objects

**Usage**

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

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

**Arguments**

x	hyperSpec object
y	not supported
use, method	handed to <a href="#">cov</a>
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  $nwl(x) \times nwl(x)$

**Author(s)**

C. Beleites

**See Also**[cov](#)**Examples**

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

---

decomposition	<i>Convert Principal Component Decomposition or the like into a hyper-Spec Object</i>
---------------	---

---

**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**

```
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, `plotspc` 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)
```

---

dimnames,hyperSpec-method

*dimnames for hyperSpec objects*

---

## Description

hyperSpec objects can have row- and column names like data.frames. The "names" of the wavelengths are treated separately: see [wl](#)

## Usage

```
## 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 <a href="#">rownames</a> or <a href="#">colnames</a> : logical. Should this create names if they are NULL?
prefix	handed to <a href="#">rownames</a> or <a href="#">colnames</a>
value	the new names

## Author(s)

C. Beleites

## See Also

[wl](#) for the wavelength dimension

[dimnames](#)

[rownames](#)

[colnames](#)

**Examples**

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

---

droplevels,hyperSpec-method  
*droplevels for hyperSpec object*

---

**Description**

calls `base::droplevels()` on the data.frame in `spc@data`.

**Usage**

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

**Arguments**

x	hyperSpec object
...	handed to <code>base::droplevels.data.frame()</code>

**Value**

hyperSpec object with unused levels of all factors in `@data` dropped.

**See Also**

`base::droplevels()`

**Examples**

```
chondro[1:3]$clusters
droplevels (chondro [1:3])$clusters
```

---

empty	<i>Empty hyperSpec object</i>
-------	-------------------------------

---

**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	<i>Quinine Fluorescence Spectra Fluorescence spectra of different dilutions of quinine forming a calibration set.</i>
-----	---

---

**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)
```

---

Future-functions	<i>Future functions</i>
------------------	-------------------------

---

**Description**

These functions will be introduced in **hyperSpec** v1.0 and will replace some current functions. Now they appear here just for compatibility with other packages, which should be released on CRAN. They are not intended to be used by **hyperSpec** v0.100 users directly.

**Usage**

```
.spc_io_postprocess_optional(...)
wl_convert_units(x, from, to, ref_wl = NULL)
hy_set_options(...)
hy_get_option(...)
hy_set_options(...)
read_txt_long(...)
read_txt_wide(...)
.wl_fix_unit_name(...)
assert_hyperSpec(...)
```

**Arguments**

```
...           Arguments to functions.
x, from, to, ref_wl
              Arguments to functions.
```

---

guess.wavelength	<i>guess wavelengths from character vector</i>
------------------	--

---

### Description

character vectors used for names (e.g. colnames for matrices or data.frames) are often treated by [make.names](#) or similar functions that produce suitable names (e.g. by prepending "X" to numbers.). Such names cannot be directly converted to numeric.

### Usage

```
guess.wavelength(X)
```

### Arguments

X                      character with numbers hidden inside

### Details

guess.wavelength tries to extract numbers from X which may be surrounded by such "protecting" characters.

### Value

numeric

### Examples

```
tmp <- data.frame(flu [[, 400 ~ 410]])
(wl <- colnames (tmp))
guess.wavelength (wl)
```

---

hy.getOptions	<i>Options for package hyperSpec Functions to access and set hyperSpec's options.</i>
---------------	---

---

### Description

Currently, the following options are defined:

Name	Default Value (range)	Description	Used by
debuglevel	0 (1L 2L 3L)	amount of debugging information produced	<a href="#">spc.identify</a> various file imp
gc	FALSE	triggers frequent calling of gc ()	<a href="#">spc.fit.poly</a> <a href="#">read.ENVI</a> , ne
file.remove.emptyspc	TRUE	remove empty spectra directly on file import	various file imp

<code>file.keep.name</code>	TRUE	always create filename column	various file imp
<code>tolerance</code>	<code>sqrt (.Machine\$double.eps)</code>	tolerance for numerical comparisons	<code>normalize01,</code>
<code>wl.tolerance</code>	<code>sqrt (.Machine\$double.eps)</code>	tolerance for comparisons of the wavelength axis	<code>all.equal, co</code>
<code>plot.spc.nmax</code>	25	number of spectra to be plotted by default	<code>plotspc</code>
<code>ggplot.spc.nmax</code>	10		<code>qplotspc</code>

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

**Author(s)**

C. Beleites

**Examples**

```
hy.getOptions ()
```

---

hy.unittest	<i>hyperSpec unit tests</i>
-------------	-----------------------------

---

**Description**

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

**Usage**

```
hy.unittest(standalone = TRUE, reporter = "progress")
```

**Arguments**

standalone	run the unit test on their own, e.g. from the console ('TRUE') or within testthat tests ('FALSE'), e.g. via 'devtools::test()'
reporter	the reporter to use, defaults to [testthat::ProgressReporter]

**Value**

Invisibly returns a data frame with the test results

**Author(s)**

Claudia Beleites

**Examples**

```
hy.unittest ()
```

---

hyperSpec-class	<i>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.</i>
-----------------	--

---

**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 "hyperspec" for an introduction to hyperSpec from a spectroscopic point of view.

**Examples**

```
showClass("hyperSpec")
## Not run: vignette("hyperspec")
```

---

initialize

*Creating a hyperSpec Object*

---

**Description**

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

**Usage**

```
## 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 <i>one</i> column of a data frame by: <code>data.frame (spc = I (as.matrix (spc)))</code> . 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, <code>1 : ncol (data\$spc)</code> 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 <a href="#">plotmath</a> ). If label is not given, a list containing NULL for each of the columns of data and wavelength is used.

### Details

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

### 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

```
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*m)))
)
```

```
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

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

## S4 method for signature 'hyperSpec'
labels(object, which = bquote(), drop = TRUE, ..., use.colnames = TRUE)
```

### Arguments

object	a hyperSpec object
which	numeric or character to specify the label(s)
...	ignored
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")
wl <- 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 (wl))
  abline (v = wl[i], col = cols[i], lwd = 2, lty = 2)

plotc (laser [, , wl], 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

**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**

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

makeraster

*makeraster*

---

**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**

```

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

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

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 acutally 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**

```
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 <a href="#">plotmap</a>
pch	symbol to display the points of the polygon for <a href="#">sel.poly</a>
size	size for polygon point symbol for <a href="#">sel.poly</a>
...	further arguments for <a href="#">grid.points</a> and <a href="#">grid.lines</a>

**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**

```
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:
```

```

plotvoronoi (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

```

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 <a href="#">text</a>
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.

### Author(s)

Claudia Beleites

### Examples

```
dend <- hclust (pearson.dist (laser[[ ]]))
par (xpd = TRUE, mar = c (5.1, 4, 4, 3)) # allows plotting into the margin
plot (dend, hang = -1, labels = FALSE)

## mark clusters
clusters <- as.factor (cutree (dend, k = 4))
levels (clusters) <- LETTERS [1 : 4]
mark.dendrogram (dend, clusters, label = "cluster")

## mark independent factor
mark.dendrogram (dend, as.factor (laser [, , 405.36] > 11000),
  pos.marker = -0.02, pos.text = - 0.03)

## mark continuous variable: convert it to a factor and omit labels
mark.dendrogram (dend, cut (laser [, , 405.36]), 100, alois.palette (100),
  pos.marker = -.015, text.col = NA,
  label = expression (I [lambda == 405.36~nm]), label.right = FALSE)
```

---

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

**Examples**

```
plot (chondro [7])
markpeak (chondro [7], 1662)
```

---

Math2,hyperSpec-method

*Math Functions for hyperSpec Objects*

---

**Description**

Mathematical functions for hyperSpec Objects.

**Usage**

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

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

## 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 "#rrbbggaa".

**Author(s)**

C. Beleites and A. Bonifacio

**See Also**

[rainbow](#)

**Examples**

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

```
plot (flu, col = matlab.dark.palette (nrow (flu)))
```

```
plotmap (chondro, col = alois.palette)
```

---

mean\_sd,numeric-method

*Mean and Standard Deviation Calculate mean and standard deviation, and mean, mean  $\pm$  one standard deviation, respectively.*

---

**Description**

These functions are provided for convenience.

**Usage**

```
## 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'
mean(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 <a href="#">mean</a> and <a href="#">sd</a>
...	ignored (needed to make function generic)
probs	the quantiles, see <a href="#">quantile</a>
names	"pretty" results in percentages (like <a href="#">quantile</a> '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$c)
mean_pm_sd (flu$spc)
mean_pm_sd (flu)
plot (mean (chondro))
plot (quantile (chondro))
```

---

merge,hyperSpec,hyperSpec-method  
*Merge hyperSpec objects*

---

## Description

Merges two hyperSpec objects and [cbinds](#) their spectra matrices, or merges additional extra data into a hyperSpec object.

## Usage

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

## S4 method for signature 'hyperSpec,data.frame'
merge(x, y, ...)

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

## Arguments

x	a hyperSpec object or data.frame
y	a hyperSpec object or data.frame (including derived classes like tibble)
...	handed to <a href="#">merge.data.frame</a>

## 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 [orderw1](#).

If a hyperSpec object and a data.frame are merged, the result is of the class of the first (x) object.

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

## merging data.frame into hyperSpec object => hyperSpec object
y <- data.frame (filename = sample (flu$filename, 4, replace = TRUE), cpred = 1:4)
y
tmp <- merge (flu, y)
tmp$.

## merging hyperSpec object into data.frame => data.frame
merge (y, flu)
```

---

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).

**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**

```
ncol (chondro)  
nrow (chondro)  
  
nwl (chondro)  
dim (chondro)  
length (chondro)
```

---

normalize01                    *normalization for mixed colors*

---

### Description

Normalize numbers -> [0, 1]

### Usage

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

[w1.eval](#), [vanderMonde](#)

---

orderwl	<i>Sorting the Wavelengths of a hyperSpec Object Rearranges the hyperSpec object so that the wavelength vector is in increasing (or decreasing) order.</i>
---------	--

---

## Description

The wavelength vector is sorted and the columns of the spectra matrix are rearranged accordingly.

## Usage

```
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.

### Format

The spectrum was acquired with a Renishaw InVia spectrometer from 100 to 3200 cm<sup>-1</sup> 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<sup>2</sup>*


---

### Description

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

### 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

**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 [plotspc](#), [plotmap](#), and [plotc](#).

**Usage**

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

**Arguments**

<code>x</code>	the hyperSpec object
<code>y</code>	selects what plot should be produced
<code>...</code>	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 `plotspc` to produce a spectra plot.

**"spcmeansd"** plots mean spectrum +/- one standard deviation

**"spcprctl5"** plots 16th, 50th, and 84th percentile spectre. If the distributions of the intensities at all wavelengths were normal, this would correspond to `"spcmeansd"`. However, this is frequently not the case. Then `"spcprctl5"` gives a better impression of the spectral data set.

**"spcprctl15"** like `"spcprctl5"`, 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

```
plot (flu)
```

```
plot (flu, "c")
```

```
plot (laser, "ts")
```

```
spc <- apply (chondro, 2, quantile, probs = 0.05)
```

```
spc <- sweep (chondro, 2, spc, "-")
```

```
plot (spc, "spcprctl5")
```

```
plot (spc, "spcprctl5")
```

```
plot (spc, "spcmeansd")
```

---

plotc	<i>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.</i>
-------	--

---

### 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.

### Usage

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

### Arguments

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

### 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**

```
## 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"))

conc <- list (c = seq (from = 0.04, to = 0.31, by = 0.01))
ci <- predict (calibration, newdata = conc, interval = "confidence", level = 0.999)

panel.ci <- function (x, y, ...,
                      conc, ci.lwr, ci.upr, ci.col = "#606060") {
  panel.xyplot (x, y, ...)
  panel.lmline (x, y,...)
  panel.lines (conc, ci.lwr, col = ci.col)
  panel.lines (conc, ci.upr, col = ci.col)
}

plotc (flu [, , 450], panel = panel.ci,
      conc = conc$c, ci.lwr = ci [, 2], ci.upr = ci [, 3])

## example 2: time-trace of laser emission modes
cols <- c ("black", "blue", "#008000", "red")
wl <- i2wl (laser, c(13, 17, 21, 23))

plotspc (laser, axis.args=list (x = list (at = seq (404.5, 405.8, .1))))
for (i in seq_along (wl))
  abline (v = wl[i], col = cols[i], lwd = 2)

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

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
)

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

**Arguments**

<code>object, data</code>	the <code>hyperSpec</code> object
<code>model, x</code>	formula specifying the columns of <code>object</code> that are to be displayed by <code>levelplot</code>
<code>func, func.args</code>	Before plotting, <code>plotmap</code> applies function <code>func</code> with the arguments given in the list <code>func.args</code> to each of the spectra. Thus a single summary value is displayed for each of the spectra. This can be suppressed manually by setting <code>func</code> to <code>NULL</code> . It is automatically suppressed if <code>.wavelength</code> appears in the formula.
<code>...</code>	further arguments are passed down the call chain, and finally to <code>levelplot</code>
<code>transform.factor</code>	If the color-coded variable is a factor, should <code>trellis.factor.key</code> be used to compute the color coding and legend?
<code>contour, useRaster</code>	see <code>levelplot</code>

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 [panel.identify](#), the indices returned by 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.

map.identify uses option debuglevel similar to [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 map.sel.\* functions offer further interactive selection, see [map.sel.poly](#).

### Value

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**

vignette (plotting), vignette (hyperspec)  
[plot](#)  
[levelplot](#)  
[trellis.factor.key](#) for improved color coding of factors  
[hyperSpec options spc.identify map.sel.poly](#)  
[panel.voronoi](#)

**Examples**

```
## Not run:
vignette (plotting)
vignette (hyperspec)

## 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 (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 <a href="#">image</a>
...	further parameters for <a href="#">image</a>
contour	should <a href="#">contour</a> be called instead of <a href="#">image</a> ?

### 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	<i>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.</i>
---------	---

---

## 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

```
plotspc(
  object,
  wl.range = TRUE,
  wl.index = FALSE,
  wl.reverse = FALSE,
  spc.nmax = hy.getOption("plot.spc.nmax"),
  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),
  debuglevel = hy.getOption("debuglevel")
)

stacked.offsets(
  x,
  stacked = TRUE,
  min.zero = FALSE,
```

```

    add.factor = 0.05,
    add.sum = 0,
    .spc = NULL,
    debuglevel = hy.getOption("debuglevel")
  )

```

## Arguments

<code>object</code>	the hyperSpec object
<code>wl.range</code>	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 <code>wl.index</code> ).
<code>wl.index</code>	if TRUE, <code>wl.range</code> is considered to give column indices into the spectra matrix. Defaults to specifying wavelength values rather than indices.
<code>wl.reverse</code>	if TRUE, the wavelength axis is plotted backwards.
<code>spc.nmax</code>	maximal number of spectra to be plotted (to avoid accidentally plotting of large numbers of spectra).
<code>func</code>	a function to apply to each wavelength in order to calculate summary spectra such as mean, min, max, etc.
<code>func.args</code>	list with further arguments for <code>func</code>
<code>stacked</code>	if not NULL, a "stacked" plot is produced, see the example. <code>stacked</code> 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.
<code>stacked.args</code>	a list with further arguments to <a href="#">stacked.offsets</a> .
<code>add</code>	if TRUE, the output is added to the existing plot
<code>bty</code>	see <a href="#">par</a>
<code>plot.args</code>	list with further arguments to <a href="#">plot</a>
<code>col</code>	see <a href="#">par</a> . <code>col</code> might be a vector giving individual colors for the spectra.
<code>lines.args</code>	list with further arguments to <a href="#">lines</a> . <code>lines.args\$type</code> defaults to "l".
<code>xoffset</code>	vector with abscissa offsets for each of the <code>wl.ranges</code> . If it has one element less than there are <code>wl.ranges</code> , 0 is padded at the beginning. 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 ~ 3200)</code> , <code>xoffset = 750</code> would result in a reasonable plot. See also the examples.
<code>yoffset</code>	ordinate offset values for the spectra. May be offsets to stack the spectra ( <a href="#">stacked.offsets</a> ). Either one for all spectra, one per spectrum or one per group in stacked.
<code>nexticks</code>	hint how many tick marks the abscissa should have.
<code>axis.args</code>	list with further arguments for <a href="#">axis</a> . <code>axis.args\$x</code> should contain arguments for plotting the abscissa, <code>axis.args\$y</code> those for the ordinate (again as lists).

break.args	list with arguments for <a href="#">axis.break</a> .
title.args	list with further arguments to <a href="#">title</a> . title.args may contain two lists, \$x, and \$y to set parameters individually for each axis.
fill	if not NULL, the area between the specified spectra is filled with color col. 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.
fill.col	character vector with fill color. Defaults to brightened colors from col.
border	character vector with border color. You will need to set the line color col to NA in order see the effect.
polygon.args	list with further arguments to <a href="#">polygon</a> which draws the filled areas.
zeroline	NA or a list with arguments <a href="#">abline</a> , 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.
debuglevel	if > 0, additional debug output is produced, see <a href="#">options</a> for details
x	a hyperSpec object
min.zero	if TRUE, the lesser of zero and the minimum intensity of the spectrum is used as minimum.
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
wavelengths	the wavelengths of the plotted spectral data points

This can be used together with [spc.identify](#).

a list containing

offsets            numeric with the yoffset for each group in stacked  
 groups            numeric with the group number for each spectrum  
 levels            if stacked is a factor, the levels of the groups

### Author(s)

C. Beileites

### 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

```
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 <a href="#">geom_point</a>
...	handed to <a href="#">geom_point</a>
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 <a href="#">geom_point</a> instead of <a href="#">ggplot</a> .

## 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")
```

---

qplotmap

*Spectra plotting with ggplot2*

---

**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 <a href="#">geom_tile</a>
...	handed to <a href="#">geom_tile</a>
func	function to summarize the wavelengths
func.args	arguments to func
map.tileonly	if TRUE, mapping will be handed to <a href="#">geom_tile</a> instead of <a href="#">ggplot</a> .

**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 ())
```

---

qplotmixmap	<i>qplotmap with colour mixing for multivariate overlay</i>
-------------	---

---

**Description**

map plot with colour overlay.

**Usage**

```
qplotmixmap(object, ...)
```

**Arguments**

object	hyperSpec object
...	handed over to <a href="#">qmixlegend</a> and <a href="#">qmixture</a>

**Value**

invisible list with ggplot2 objects map and legend

**Author(s)**

Claudia Beleites

**See Also**[qmixture](#)**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"))

```

qplotspc

*Spectra plotting with ggplot2***Description**

Spectra plotting with ggplot2

**Usage**

```

qplotspc(
  x,
  wl.range = TRUE,
  ...,
  mapping = aes_string(x = ".wavelength", y = "spc", group = ".rownames"),
  spc.nmax = hy.getOption("ggplot.spc.nmax"),
  map.lineonly = FALSE,
  debuglevel = hy.getOption("debuglevel")
)

```

**Arguments**

x	hyperSpec object
wl.range	wavelength ranges to plot
...	handed to <a href="#">geom_line</a>
mapping	see <a href="#">geom_line</a>
spc.nmax	maximum number of spectra to plot
map.lineonly	if TRUE, mapping will be handed to <a href="#">geom_line</a> instead of <a href="#">ggplot</a> .
debuglevel	if > 0, additional debug output is produced

**Details**

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

**Value**

a [ggplot](#) object

**Author(s)**

Claudia Beleites

**See Also**

[plotspc](#)

[ggplotgeom\\_line](#)

**Examples**

```
plotspc (chondro)
```

```
plotspc (paracetamol, c (2800 ~ max, min ~ 1800)) + scale_x_reverse (breaks = seq (0, 3200, 400))
```

```
plotspc (aggregate (chondro, chondro$clusters, mean),
         mapping = aes (x = .wavelength, y = spc, colour = clusters)) +
  facet_grid (clusters ~ .)
```

```
plotspc (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 (via `make.names(unique = TRUE)`).

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**

```
## S3 method for class 'matrix'
rbind.fill(...)
```

```
## S3 method for class 'fill'
rbind(...)
```

**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

**Author(s)**

C. Beleites

**See Also**

[rbind](#), [cbind](#), [plyr::rbind.fill\(\)](#)

**Examples**

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

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

hyperSpec:::rbind.fill.matrix (A, 99)

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

---

read.asc.PerkinElmer *File import filter PerkinElmer ASCII spectra*

---

**Description**

Imports a single spectrum in PerkinElmer's ASCII format. This function is experimental.

**Usage**

```
read.asc.PerkinElmer(file = stop("filename or connection needed"), ...)
```

**Arguments**

file filename (or connection)  
 ... further parameters are handed to [read.txt.long](#)

**Value**

hyperSpec object

---

read.cytomat *Import for Cytospec mat files*

---

**Description**

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

**Usage**

```
read.cytomat(...)
```

```
read.mat.Cytospec(file, keys2data = FALSE, blocks = TRUE)
```

**Arguments**

... read.cytomat for now hands all arguments to read.mat.Cytospec for backwards compatibility.  
 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.

**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

**See Also**

R.matlab::readMat

---

read.ENVI

*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(
  file = stop("read.ENVI: file name needed"),
  headerfile = NULL,
  header = list(),
  keys.hdr2data = FALSE,
  x = 0:1,
  y = x,
  wavelength = NULL,
  label = list(),
  block.lines.skip = 0,
  block.lines.size = NULL,
  ...,
  pull.header.lines = TRUE
)

read.ENVI.HySpex(
  file = stop("read.ENVI.HySpex: file name needed"),
  headerfile = NULL,
  header = list(),
  keys.hdr2data = NULL,
  ...
)

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

**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 as file but hdr or HDR suffix.
header	list with header information, see details. Overwrites information extracted from the header file.
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.
x, y	vectors of form c(offset, step size) for the position vectors, see details.
wavelength, label	lists that overwrite the respective information from the ENVI header file. These data is then handed to <code>initialize</code>
block.lines.skip, block.lines.size	BIL and BIP ENVI files may be read in blocks of lines: skip the first <code>block.lines.skip</code> lines, then read a block of <code>block.lines.size</code> lines. If <code>block.lines.NULL</code> , the whole file is read. Blocks are silently truncated at the end of the file (more precisely: to <code>header\$lines</code> ).
...	currently unused by <code>read.ENVI</code> , <code>read.ENVI.Nicolet</code> hands those arguments over to <code>read.ENVI</code>
pull.header.lines	(internal) flag whether multi-line header entries grouped by curly braces should be pulled into one line each.
nicolet.correction	see details

**Details**

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:

header\$	values	meaning
samples	integer	no of columns / spectra in x direction
lines	integer	no of lines / spectra in y direction
bands	integer	no of wavelengths / data points per spectrum
'data type'		format of the binary file
	1	1 byte unsigned integer
	2	2 byte signed integer
	3	4 byte signed integer
	4	4 byte float
	5	8 byte double

	9	16 (2 x 8) byte complex double
	12	2 byte unsigned integer
'header offset'	integer	number of bytes to skip before binary data starts
interleave		directions of the data cube
	"BSQ"	band sequential (indexing: [sample, line, band])
	"BIL"	band interleave by line (indexing: [sample, line, band])
	"BIP"	band interleave by pixel (indexing: [band, line, sample])
'byte order'	0 or "little"	little endian
	1 or "big"	big endian
	"swap"	swap byte order

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 header\$*samples* - 1 and header\$*lines* - 1, respectively. *x* and *y* give offset of the first spectrum and step size.

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

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 <i>x</i> and <i>y</i> step size (specify <i>x</i> = NA and <i>y</i> = 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.

## Value

a hyperSpec object

## Functions

- `read.ENVI.HySpex()`:
- `read.ENVI.Nicolet()`:

## Author(s)

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

## References

This function was adapted from `caTools::read.ENVI()`:

Jarek Tuszynski (2008). *caTools: Tools: moving window statistics, GIF, Base64, ROC AUC, etc..* R package version 1.9.

**See Also**

caTools::read.ENVI()  
[textio](#)

---

read.ini                      *Read INI files*

---

**Description**

read.ini reads ini files of the form

**Usage**

```
read.ini(  
  con = stop("Connection con needed."),  
  skip = NULL,  
  encoding = "unknown"  
)
```

**Arguments**

con	connection or file name
skip	number of lines to skip before first [section] starts
encoding	see <a href="#">readLines</a>

**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

read.jdx

*JCAMP-DX Import for Shimadzu Library Spectra***Description**

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

**Usage**

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

**Arguments**

filename	file name and path of the .jdx file
encoding	encoding of the JCAMP-DX file (used by <code>base::readLines()</code> )
header	list with manually set header values
keys.hdr2data	index vector indicating which header entries should be tranfered into the extra data. Usually a character vector of labels (lowercase, without and dashes, blanks, underscores). If TRUE, all header entries are read.
...	further parameters handed to the data import function, e.g.

parameter	meaning	default
xtol	tolerance for checking calculated x values against checkpoints at beginning of line	XFACTOR
ytol	tolerance for checking Y values against MINY and MAXY	YFACTOR

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 hyperSpec object (if FALSE, a list of hyperSpec objects is returned).
wl.tolerance, collapse.equal	see <a href="#">collapse</a>

**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.

**Author(s)**

C. Beleites with contributions by Bryan Hanson

---

read.spc

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

---

**Description**

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

**Usage**

```
read.spc(  
  filename,  
  keys.hdr2data = FALSE,  
  keys.log2data = 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.log2data	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. All header fields specified in the .spc file format specification (see below) are imported and can be referred to by their de-capitalized names.
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**

```
## 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	<i>read Kaiser .spc files</i>
-----------------	-------------------------------

---

**Description**

Import functions for Kaiser Optical Systems .spc files

**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 <a href="#">read.spc</a> .
type	what kind of measurement was done? If "map", read.spc.KaiserMap is used instead of read.spc.Kaiser.

**Details**

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

**Value**

hyperSpec

**Examples**

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

---

read.spe	<i>Import WinSpec SPE file</i>
----------	--------------------------------

---

**Description**

Import function for WinSpec SPE files (file version up to 3.0). 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**

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

.read.spe.header(filename)

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 <i>accumulated</i> spectra to <i>averaged</i> 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

---

read.txt.Horiba      *Import Horiba Labspec exported ASCII files*

---

**Description**

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

**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

---

read.txt.Shimadzu	<i>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)</i>
-------------------	--

---

### 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

```
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

---

read.txt.wide	<i>Import/export of hyperSpec objects to/from ASCII files A detailed discussion of hyperSpec's file import and export capabilities is given in vignette "fileio".</i>
---------------	---

---

**Description**

Besides [save](#) and [load](#), two general ways to import and export data into hyperSpec objects exist.

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

**Usage**

```
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,
  ...
)

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

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.
check.names	handed to <a href="#">read.table</a> . Make sure this is FALSE, if the column names of the spectra are the wavelength values.
...	arguments handed to <a href="#">read.table</a> and <a href="#">write.table</a> , respectively.
header	the file has (shall have) a header line
object	the hyperSpec object
order	which columns should be <a href="#">ordered</a> ? order is used as index vector into a data.frame with columns given by cols.
na.last	handed to <a href="#">order</a> by <a href="#">write.txt.long</a> .
decreasing	logical vector giving the sort order
quote, sep, col.names, row.names	have their usual meaning (see <a href="#">read.table</a> and <a href="#">write.table</a> ), 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 <a href="#">write.txt.wide</a>

## Details

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

A second option is using the package [R.matlab](#) which provides the functions [readMat](#) and [writeMat](#). hyperSpec comes with a number of pre-defined functions to import manufacturer specific file formats. For details, see vignette ("fileio").

[read.spc](#) imports Thermo Galactic's .spc file format, and ENVI files may be read using [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 **hyperSpec**. An example is in the "flu" vignette (see vignette ("flu", package = "hyperSpec").

Note that R accepts many packed formats for ASCII files, see [connections](#). For .zip files, see [unzip](#).

For further information, see the examples below, vignette ("fileio") and the documentation of [R.matlab](#).

A second option is using the package [R.matlab](#) which provides the functions [readMat](#) and [writeMat](#). hyperSpec comes with a number of pre-defined functions to import manufacturer specific file formats. For details, see vignette ("file-io").

[read.spc](#) imports Thermo Galactic's .spc file format, and ENVI files may be read using [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 **hyperSpec**. An example is in the "flu" vignette (see vignette ("flu", package = "hyperSpec").

Note that R accepts many packed formats for ASCII files, see [connections](#). For .zip files, see [unzip](#).

For further information, see the examples below and the documentation of [R.matlab](#).

## Author(s)

C. Beleites

## See Also

vignette ("fileio") and <http://hyperspec.r-forge.r-project.org/blob/fileio.pdf>, respectively

[read.table](#) and [write.table](#)

[R.matlab](#) for .mat files

[read.ENVI](#) for ENVI data

[read.spc](#) for .spc files

Manufacturer specific file formats: [read.txt.Renishaw](#)

**Examples**

```

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

## export & import matlab files
if (require (R.matlab)) {
  # export to matlab file
  writeMat (paste0 (tempdir(), "/test.mat"),
            x = flu[[[]], wavelength = flu@wavelength,
              label = lapply (flu@label, as.character))

  # reading a matlab file
  data <- readMat (paste0 (tempdir(), "/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 = paste0 (tempdir(), "/flu.txt"),
               cols = c(".wavelength", "spc", "c"),
               order = c("c", ".wavelength"),
               decreasing = c(FALSE, TRUE))

read.txt.long (file = paste0 (tempdir(), "/flu.txt"),
              cols = list (.wavelength = expression (lambda / nm),
                          spc = "I / a.u", c = expression ("/" (c, (mg/l))))))

write.txt.wide (flu, file = paste0 (tempdir(), "/flu.txt"),
               cols = c("c", "spc"),
               col.labels = TRUE, header.lines = 2, row.names = TRUE)

write.txt.wide (flu, file = paste0 (tempdir(), "/flu.txt"),
               col.labels = FALSE, row.names = FALSE)

read.txt.wide (file = paste0 (tempdir(), "/flu.txt"),
              # give columns in same order as they are in the file
              cols = list (spc = "I / a.u",
                          c = expression ("/" ("c", "mg/l")),
                          filename = "filename",
                          # plus wavelength label last
                          .wavelength = "lambda / nm"),
              header = TRUE)

```

rmmvnorm

*Multivariate normal random numbers***Description**

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

**Usage**

```
rmmvnorm(n, mean, sigma)

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

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

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

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

**Arguments**

<code>n</code>	vector giving the numer of cases to generate for each group
<code>mean</code>	matrix with mean cases in rows
<code>sigma</code>	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)
  }

```

---

sample,hyperSpec-method

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

---

## 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

```

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

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

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

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

```

## Arguments

x	The hyperSpec object, data.frame or matrix 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 <a href="#">drop</a> : 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 <a href="#">as.matrix</a> (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 <a href="#">as.matrix</a> (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)
```

---

seq.hyperSpec

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

---

**Description**

Note that [wl2i](#) generates sequences of indices along the wavelength axis.

**Usage**

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

**Arguments**

x	the hyperSpec object
from, to	arguments handed to <a href="#">seq.int</a>
...	arguments for <a href="#">seq</a> , namely by, length.out
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 ... arguments (on which no dispatch with differing types is possible).

[seq\\_along](#) is not generic, but returns a sequence of the length of the object. As `hyperSpec` provides a Method [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.

[seq](#)

**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**

```
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`.

### Usage

```
spc.fit.poly(
  fit.to,
  apply.to = NULL,
  poly.order = 1,
  offset.wl = !(is.null(apply.to))
)

spc.fit.poly.below(
  fit.to,
  apply.to = fit.to,
  poly.order = 1,
  npts.min = max(round(nwl(fit.to) * 0.05), 3 * (poly.order + 1)),
  noise = 0,
  offset.wl = FALSE,
  max.iter = nwl(fit.to),
  stop.on.increase = FALSE,
  debuglevel = hy.getOption("debuglevel")
)
```

### Arguments

<code>fit.to</code>	hyperSpec object on which the baselines are fitted
<code>apply.to</code>	hyperSpec object on which the baselines are evaluated If NULL, a hyperSpec object containing the polynomial coefficients rather than evaluated baselines is returned.
<code>poly.order</code>	order of the polynomial to be used
<code>offset.wl</code>	should the wavelength range be mapped to $\rightarrow [0, \text{delta wl}]$ ? This enhances numerical stability.
<code>npts.min</code>	minimal number of points used for fitting the polynomial
<code>noise</code>	noise level to be considered during the fit. It may be given as one value for all the spectra, or for each spectrum separately.
<code>max.iter</code>	stop at the latest after so many iterations.
<code>stop.on.increase</code>	additional stopping rule: stop if the number of support points would increase, regardless whether <code>npts.min</code> was reached or not.

`debuglevel` additional output: 1 shows `npts.min`, 2 plots support points for the final baseline of 1st spectrum, 3 plots support points for 1st spectrum, 4 plots support points for all spectra.

### 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")`  
see [options](#) for more on `debuglevel`

### Examples

```
## 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.fit.poly.below(chondro [1:3], debuglevel = 1)
spc.fit.poly.below(chondro [1:3], debuglevel = 2)
spc.fit.poly.below(chondro [1:3], debuglevel = 3, noise = sqrt (rowMeans (chondro [[1:3]])))
```

---

<code>spc.identify</code>	<i>Identifying Spectra and Spectral Data Points This function allows to identify the spectrum and the wavelength of a point in a plot produced by <a href="#">plotspc</a>.</i>
---------------------------	--

---

### 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, wlclic)

spc.point.default(wl, spc, wlclic)

spc.point.min(wl, spc, wlclic)

spc.point.sqr(wl, spc, wlclic, 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 <code>plotspc</code>
y	the ordinate values. Giving y will override any values from <code>x\$y</code> .
wavelengths	the wavelengths for the data points. Giving wavelengths will override any values from <code>x\$wavelengths</code> .
ispc	if a selection of spectra was plotted, their indices can be given in <code>ispc</code> . In this case <code>ispc [i]</code> is returned rather than <code>i</code> .
tol.wl, tol.spc	tolerance in wavelength and spectral intensity to search around the clicked point. See details.
point.fn	function ( <code>wl, spc, wlclic</code> ) to determine the actual point to label, see details.
formatter	function ( <code>i, wl, spc</code> ) that produces the labels. If NULL, no labels are displayed.
...	passed to <code>text</code> in order to produce the labels
cex, adj, srt	see <code>par</code>

warn	Should the user be warned if no point is in the considered window? In addition, see the discussion of option <code>debugLevel</code> in the details. If FALSE, the resulting <code>data.frame</code> will have a row of NAs instead.
wl	the wavelength to label
spc	the intensity to label
wlclick	the clicked wavelength
delta	<code>spc.point.sqr</code> fits the parabola in the window <code>wlclick ± delta</code> points.
digits	how many digits of the wavelength should be displayed?

### Details

In a second step, `max.fn` 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:

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

`point.fn` 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:

<code>spc.label.default</code>	spectrum number, wavelength
<code>spc.label.wlonly</code>	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 off 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](#)

[map.identify](#) [map.sel.poly](#)

**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	<i>loess smoothing interpolation for spectra Spectra can be smoothed and interpolated on a new wavelength axis using <a href="#">loess</a>.</i>
-----------	---

---

### 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

```
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 <a href="#">loess</a> and <a href="#">loess.control</a> .

### Value

a new hyper spec object.

### Author(s)

C. Beleites

### See Also

[loess](#), [loess.control](#)

### Examples

```
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.approx                    *Impute missing data points*

---

### Description

Replace NAs in the spectra matrix by interpolation. With less than 4 points available linear interpolation of the 2 neighbour points is used. For larger numbers of neighbour points, smoothing interpolation is performed by [smooth.spline](#).

### Usage

```
spc.NA.approx(
  spc,
  neighbours = 1,
  w = rep(1, 2 * neighbours),
  df = 1 + .Machine$double.eps,
  spar = NULL,
  debuglevel = hy.getOption("debuglevel")
)

spc.NA.linapprox(...)
```

### Arguments

spc	hyperSpec object with spectra matrix containing NAs
neighbours	how many neighbour data points should be used to fit the line
w, df, spar	see <a href="#">smooth.spline</a>
debuglevel	see <a href="#">options</a>
...	ignored

### Value

hyperSpec object

### Note

The function has been renamed from spc.NA.linapprox to spc.NA.approx

### Author(s)

Claudia Beleites

### Examples

```
fluNA <- hyperSpec:::fluNA
spc.NA.approx (fluNA [, , min ~ 410], debuglevel = 1)
spc.NA.approx (fluNA [1, , min ~ 410], debuglevel = 2)
spc.NA.approx (fluNA [4, , min ~ 410], neighbours = 3, df = 4, debuglevel = 2)
```

---

spc.rubberband	<i>Rubberband baseline correction</i>
----------------	---------------------------------------

---

**Description**

Rubberband baseline

**Usage**

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

**Arguments**

spc	hyperSpec object
...	further parameters handed to <a href="#">smooth.spline</a>
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 >= 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 ~ 1800])
bl <- spc.rubberband (paracetamol [, , 175 ~ 1800], noise = 300, df = 20)
plot (bl, add = TRUE, col = 2)

plot (paracetamol [, , 175 ~ 1800] - bl)
```

---

spc.smooth.spline      *Spectral smoothing by splines*

---

**Description**

Smoothing splines

**Usage**

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

**Arguments**

spc	hyperSpec object
newx	wavelength axis to interpolate on
...	further parameters handed to <a href="#">smooth.spline</a>

**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	<i>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.</i>
-------	---

---

**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)

```

**Arguments**

x	the hyperSpec object
f	a factor giving the grouping (or a variable that can be converted into a factor by <code>as.factor</code> )
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**

```
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	<i>subset</i>
--------	---------------

---

**Description**

subset for hyperSpec object

**Usage**

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

**Arguments**

x	hyperSpec object
...	handed to <a href="#">subset</a> (data.frame method)

**Value**

hyperSpec object containing the respective subset of spectra.

**Author(s)**

Claudia Beleites

**See Also**

[subset](#)

---

Summary

*The functions*

---

## Description

all, any,

## Usage

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

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

```
all_wl(expression, na.rm = FALSE)
```

```
any_wl(expression, na.rm = FALSE)
```

## Arguments

x	hyperSpec object
...	further objects
na.rm	logical indicating whether missing values should be removed
expression	expression that evaluates to a logical matrix of the same size as the spectra matrix

## 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]);

all_wl (flu > 100)

any_wl (flu > 300)
! any_wl (is.na (flu))

```

---

sweep	<i>Sweep Summary Statistic out of an hyperSpec Object</i> <code>sweep</code> 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 <i>know</i> that dimensions match.
...	further arguments for FUN

**Details**

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

**Value**

A hyperSpec object.

**Author(s)**

C. Beleites

**See Also**[sweep](#)**Examples**

```
## Subtract 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 subtracted
background <- apply (chondro, 2, quantile, probs = 0.05)
corrected <- sweep (chondro, 2, background, "-")
plot (corrected, "spcprct15")

## Offset correction
offsets <- apply (chondro, 1, min)
corrected <- sweep (chondro, 1, offsets, "-")
plot (corrected, "spcprct15")

## 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, "spcprct15")

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

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

---

trellis.factor.key	<i>Color coding legend for factors Modifies a list of lattice arguments (as for levelplot, etc.) according to the factor levels. The colorkey will show all levels (including unused), and the drawing colors will be set accordingly.</i>
--------------------	--

---

**Description**

trellis.factor.key is used during levelplot-based plotting of factors (for hyperSpec objects) unless transform.factor = FALSE is specified.

**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"))
```

---

 vanderMonde

*Function evaluation on hyperSpec objects*


---

**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**

<code>x</code>	object to evaluate the polynomial on
<code>order</code>	of the polynomial
<code>...</code>	hyperSpec method: further arguments to <a href="#">decomposition</a>
<code>normalize.wl</code>	function to transform the wavelengths before evaluating the polynomial (or other function). <a href="#">normalize01</a> maps the wavelength range to the interval [0, 1]. Use <a href="#">I</a> 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

*line/word/character count of ASCII files*

---

**Description**

`'wc()'` is defunct and will be removed from hyperSpec in future. Consider using `[count_lines()]` instead for line counting.

**Usage**

```
wc()
```

**Author(s)**

C. Beleites

**See Also**`[count_lines()]`


---

<code>wl</code>	<i>Getting and Setting the Wavelength Axis</i> <code>wl</code> returns the wavelength axis, <code>wl&lt;-</code> 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**

<code>x</code>	a <code>hyperSpec</code> object
<code>label</code>	The label for the new wavelength axis. See <a href="#">initialize</a> for details.
<code>digits</code>	handed to <a href="#">signif</a> . See details.
<code>value</code>	either a numeric containing the new wavelength vector, or a list with <code>value\$wl</code> containing the new wavelength vector and <code>value\$label</code> holding the corresponding label.

**Details**

When the wavelength axis is replaced, the `colnames` of `x@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**

```
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 / nm))
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 <a href="#">normalize01</a> 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)))
```

---

w12i	<i>Conversion between Wavelength and Spectra Matrix Column Index</i> w12i 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**

```
w12i(x, wavelength = stop("wavelengths are required."), unlist = TRUE)
```

```
i2wl(x, i)
```

**Arguments**

x	a hyperSpec object
wavelength	the wavelengths to be converted into column indices, either numeric or a formula, see details.
unlist	if multiple wavelength ranges are given, should the indices be unlisted or kept in a list?
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 `orderw1` 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 `w12i`  
`i2w1` returns a numeric with the wavelengths

**Author(s)**

C. Beleites

**Examples**

```
flu
w12i (flu, 405 : 407)
w12i (flu, 405 ~ 407)

## beginning of the spectrum to 407 nm
w12i (flu, min ~ 407)

## 2 data points from the beginning of the spectrum to 407 nm
w12i (flu, min + 2i ~ 407)

## the first 3 data points
w12i (flu, min ~ min + 2i)

## from 490 nm to end of the spectrum
w12i (flu, 490 ~ max)

## the last 8 data points
w12i (flu, max - 7i ~ max)

## get 450 nm +- 3 data points
w12i (flu, 450 - 3i ~ 450 + 3i)

w12i (flu, 300 : 400) ## all NA:
w12i (flu, 600 ~ 700) ## NULL: completely outside flu's wavelength range

i2w1 (chondro, 17:20)
```

---

wlconv	<i>Convert different wavelength units</i>
--------	---

---

**Description**

The following units can be converted into each other: *nm*,  $cm^{-1}$ , *eV*, *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 **nm** -> **frequency in THz**
- invcm2raman(): conversion **inverse cm (absolute wavenumber)** -> **Raman shift (relative wavenumber)**
- invcm2nm(): conversion **inverse cm (absolute wavenumber)** -> **nanometers**
- invcm2ev(): conversion **inverse cm (absolute wavenumber)** -> **electronvolt**
- invcm2freq(): conversion **inverse cm (absolute wavenumber)** -> **frequency in THz**
- raman2invcm(): conversion **Raman shift (relative wavenumber)** -> **inverse cm (absolute wavenumber)**
- raman2nm(): conversion **Raman shift (relative wavenumber)** -> **nanometers**
- raman2ev(): conversion **Raman shift (relative wavenumber)** -> **electronvolt**
- raman2freq(): conversion **Raman shift (relative wavenumber)** -> **frequency in THz**
- ev2raman(): conversion **electronvolt** -> **Raman shift (relative wavenumber)**
- ev2invcm(): conversion **electronvolt** -> **inverse cm (absolute wavenumber)**
- ev2nm(): conversion **electronvolt** -> **nanometers**
- ev2freq(): conversion **electronvolt** -> **frequency in THz**
- freq2nm(): conversion **frequency in THz** -> **nanometers**
- freq2invcm(): conversion **frequency in THz** -> **inverse cm (absolute wavenumber)**
- freq2ev(): conversion **frequency in THz** -> **electronvolt**
- freq2raman(): conversion **frequency in THz** -> **Raman shift (relative wavenumber)**

### Author(s)

R. Kiselev

**Examples**

```
wlconv (3200, "Raman shift", "nm", laser = 785.04)
wlconv( 785, "nm", "invcm")
```

---

[,hyperSpec-method      *Extract and Replace parts of hyperSpec objects*

---

**Description**

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

**Usage**

```
## 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 [[<- 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 x@data
l	selecting columns of the spectra matrix. If l is numeric, the default behaviour is treating l as wavelengths, <i>not</i> 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 <a href="#">wl2i</a> .

drop	For [[]: drop unnecessary dimensions, see <a href="#">drop</a> and <a href="#">Extract</a> . 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, \dots] <- value
```

```
## S4 method for signature 'hyperSpec':
```

```
x [[i, j, l, wl.index = FALSE, \dots]] <- 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**

```
## 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
plot (flu, "spc", lines.args = list (lty = 2))
index <- rnorm (6) > 0
index
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)])
chondro$x

## 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.")
```

# Index

- !=, hyperSpec, hyperSpec-method (Comparison), 25
- \* **IO**
  - read.cytomat, 77
  - read.ENVI, 78
  - read.ini, 81
  - read.spc, 83
  - read.txt.wide, 89
- \* **aplot**
  - trellis.factor.key, 112
- \* **arith**
  - Arith, 12
  - Comparison, 25
- \* **array**
  - aggregate, 8
- \* **baseline**
  - spc.fit.poly, 99
- \* **category**
  - aggregate, 8
- \* **classes**
  - hyperSpec-class, 37
- \* **cluster**
  - pearson.dist, 58
- \* **color**
  - matlab.palette, 50
- \* **datagen**
  - initialize, 38
  - spc.bin, 97
  - spc.fit.poly, 99
  - spc.loess, 104
- \* **datasets**
  - barbiturates, 18
  - chondro, 22
  - flu, 33
  - laser, 41
  - paracetamol, 58
- \* **distribution**
  - sample, hyperSpec-method, 94
- \* **file**
  - read.cytomat, 77
  - read.ENVI, 78
  - read.ini, 81
  - read.spc, 83
  - read.txt.wide, 89
- \* **hplot**
  - plot-methods, 59
  - plotc, 61
  - plotmap, 62
  - plotspc, 67
- \* **hyperSpec arithmetical operators**
  - Arith, 12
- \* **hyperSpec arithmetic**
  - Arith, 12
- \* **hyperSpec division**
  - Arith, 12
- \* **hyperSpec matrix multiplication**
  - Arith, 12
- \* **hyperSpec plus**
  - Arith, 12
- \* **hyperSpec spectra conversion**
  - Arith, 12
- \* **iplot**
  - map.sel.poly, 45
  - spc.identify, 100
- \* **iteration**
  - apply, 10
- \* **manip**
  - [, hyperSpec-method, 121
  - bind, 19
  - collapse, 23
  - decomposition, 29
  - empty, 33
  - merge, hyperSpec, hyperSpec-method, 53
  - rbind.fill.matrix, 75
  - seq.hyperSpec, 96
  - spc.bin, 97
  - spc.fit.poly, 99

- spc.loess, 104
- \* **math**
  - Math2, hyperSpec-method, 49
- \* **methods**
  - [, hyperSpec-method, 121
  - aggregate, 8
  - apply, 10
  - Arith, 12
  - as.character, hyperSpec-method, 14
  - as.data.frame, 15
  - bind, 19
  - chk.hy, 21
  - Comparison, 25
  - decomposition, 29
  - dimnames, hyperSpec-method, 31
  - initialize, 38
  - Math2, hyperSpec-method, 49
  - ncol, hyperSpec-method, 54
  - plot-methods, 59
  - sample, hyperSpec-method, 94
  - scale, hyperSpec-method, 95
  - split, 108
  - sweep, 111
- \* **misc**
  - hy.getOptions, 35
- \* **multivar**
  - mean\_sd, numeric-method, 51
- \* **print**
  - as.character, hyperSpec-method, 14
- \* **programming**
  - hy.unittest, 37
- \* **univar**
  - mean\_sd, numeric-method, 51
- \* **utilities**
  - .DollarNames.hyperSpec, 6
  - hy.unittest, 37
- \*, hyperSpec, hyperSpec-method (Arith), 12
- +, hyperSpec, hyperSpec-method (Arith), 12
- , hyperSpec, hyperSpec-method (Arith), 12
- .DollarNames, 7
- .DollarNames (.DollarNames.hyperSpec), 6
- .DollarNames, hyperSpec-method
  - (.DollarNames.hyperSpec), 6
- .DollarNames.hyperSpec, 6
- .cluster.wavelengths, 5
- .collapse.equal, 6
- .fix\_spc\_colnames, 7
- .read.spe.header (read.spe), 86
- .read.spe.xml, 7
- .read.spe.xml\_string, 8
- .spc\_io\_postprocess\_optional
  - (Future-functions), 34
- .wl\_fix\_unit\_name (Future-functions), 34
- /, hyperSpec, hyperSpec-method (Arith), 12
- <, hyperSpec, hyperSpec-method
  - (Comparison), 25
- <=, hyperSpec, hyperSpec-method
  - (Comparison), 25
- ==, hyperSpec, hyperSpec-method
  - (Comparison), 25
- >, hyperSpec, hyperSpec-method
  - (Comparison), 25
- >=, hyperSpec, hyperSpec-method
  - (Comparison), 25
- [, 116
- [ ([, hyperSpec-method), 121
- [, hyperSpec-method, 121
- [<- ([, hyperSpec-method), 121
- [<-, hyperSpec-method
  - ([, hyperSpec-method), 121
- [[ ([, hyperSpec-method), 121
- [[, hyperSpec-method
  - ([, hyperSpec-method), 121
- [[<- ([, hyperSpec-method), 121
- [[<-, hyperSpec-method
  - ([, hyperSpec-method), 121
- \$([, hyperSpec-method), 121
- \$, hyperSpec-method
  - ([, hyperSpec-method), 121
- \$<- ([, hyperSpec-method), 121
- \$<-, hyperSpec-method
  - ([, hyperSpec-method), 121
- %\*% (Arith), 12
- %\*%, hyperSpec, hyperSpec-method (Arith), 12
- %\*%, hyperSpec, matrix-method (Arith), 12
- %\*%, matrix, hyperSpec-method (Arith), 12
- %/%, hyperSpec, hyperSpec-method (Arith), 12
- %, hyperSpec, hyperSpec-method (Arith), 12
- %\*%, 30
- ^, hyperSpec, hyperSpec-method (Arith), 12
- abline, 69
- abs, hyperSpec-method
  - (Math2, hyperSpec-method), 49

- acos, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- acosh, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- aggregate, 8, 9, 11
- aggregate, hyperSpec-method (aggregate), 8
- all, hyperSpec-method (Summary), 110
- all.equal, 26, 27, 36, 110
- all.equal (Comparison), 25
- all.equal, hyperSpec, hyperSpec-method (Comparison), 25
- all\_wl (Summary), 110
- alois.palette (matlab.palette), 50
- any, hyperSpec-method (Summary), 110
- any\_wl (Summary), 110
- apply, 10, 10, 11
- apply, hyperSpec-method (apply), 10
- Arith, 12, 27, 50
- Arith, hyperSpec, hyperSpec-method (Arith), 12
- Arith, hyperSpec, matrix-method (Arith), 12
- Arith, hyperSpec, missing-method (Arith), 12
- Arith, hyperSpec, numeric-method (Arith), 12
- Arith, hyperSpec-method (Arith), 12
- Arith, matrix, hyperSpec-method (Arith), 12
- Arith, numeric, hyperSpec-method (Arith), 12
- Arithmetic, 13, 14
- as.character, 15
- as.character
  - (as.character, hyperSpec-method), 14
- as.character, hyperSpec-method, 14
- as.data.frame, 13, 15, 16
- as.data.frame, hyperSpec-method (as.data.frame), 15
- as.data.frame.hyperSpec (as.data.frame), 15
- as.dist, 59
- as.hyperSpec, 17
- as.hyperSpec, data.frame-method (as.hyperSpec), 17
- as.hyperSpec, matrix-method (as.hyperSpec), 17
- as.long.df (as.data.frame), 15
- as.matrix, 95
- as.matrix (as.data.frame), 15
- as.matrix, hyperSpec-method (as.data.frame), 15
- as.matrix.hyperSpec (as.data.frame), 15
- as.t.df (as.data.frame), 15
- as.wide.df (as.data.frame), 15
- asin, hyperSpec-method (Math2, hyperSpec-method), 49
- asinh, hyperSpec-method (Math2, hyperSpec-method), 49
- assert\_hyperSpec (Future-functions), 34
- atan, hyperSpec-method (Math2, hyperSpec-method), 49
- atanh, hyperSpec-method (Math2, hyperSpec-method), 49
- ave, 9
- ave, hyperSpec-method (aggregate), 8
- axis, 68–70
- axis.break, 69, 70
- barbiturates, 18
- base::as.matrix(), 16
- base::droplevels(), 32
- base::droplevels.data.frame(), 32
- base::readLines(), 82
- bind, 19, 108
- cbind, 20, 53, 54, 76
- cbind.hyperSpec (bind), 19
- cbind2, 20
- cbind2, hyperSpec, hyperSpec-method (bind), 19
- cbind2, hyperSpec, missing-method (bind), 19
- ceiling, hyperSpec-method (Math2, hyperSpec-method), 49
- chk.hy, 21
- chondro, 22, 64
- collapse, 20, 23, 36, 54, 82
- colMeans, 24
- colMeans, hyperSpec-method (colSums), 24
- colmix.rgb, 43
- colmix.rgb (legendright), 42
- colnames, 31
- colnames (dimnames, hyperSpec-method), 31

- colnames, hyperSpec-method
  - (dimnames, hyperSpec-method), 31
- colnames<- (dimnames, hyperSpec-method), 31
- colnames<- , hyperSpec-method
  - (dimnames, hyperSpec-method), 31
- colorRampPalette, 50
- colSums, 24, 24, 25
- colSums, hyperSpec-method (colSums), 24
- Compare, hyperSpec, hyperSpec-method
  - (Comparison), 25
- Compare, hyperSpec, matrix-method
  - (Comparison), 25
- Compare, hyperSpec, numeric-method
  - (Comparison), 25
- Compare, hyperSpec-method (Comparison), 25
- Compare, matrix, hyperSpec-method
  - (Comparison), 25
- Compare, numeric, hyperSpec-method
  - (Comparison), 25
- Comparison, 14, 25, 26, 27, 50
- connections, 91
- contour, 66
- cos, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- cosh, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- count\_lines, 27
- cov, 28, 29, 93
- cov, hyperSpec, missing-method, 28
- create(initialize), 38
- create, hyperSpec-method (initialize), 38
- cummax, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- cummin, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- cumprod, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- cumsum, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- decomposition, 29, 114
- digamma, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- dim, 55
- dim, hyperSpec-method
  - (ncol, hyperSpec-method), 54
- dimnames, 31
- dimnames (dimnames, hyperSpec-method), 31
- dimnames, hyperSpec-method, 31
- do.call, 20
- drop, 94, 122, 123
- droplevels, hyperSpec-method, 32
- empty, 33
- ev2freq (wlconv), 119
- ev2invcm (wlconv), 119
- ev2nm (wlconv), 119
- ev2raman (wlconv), 119
- exp, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- expm1, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- export (read.txt.wide), 89
- Extract, 122, 123
- fitraster (makeraster), 44
- floor, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- flu, 33
- freq2ev (wlconv), 119
- freq2invcm (wlconv), 119
- freq2nm (wlconv), 119
- freq2raman (wlconv), 119
- Future-functions, 34
- gamma, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- geom\_line, 74, 75
- geom\_point, 71, 72
- geom\_tile, 43, 72, 73
- ggplot, 43, 71–75
- grid.lines, 46
- grid.locator, 45, 46
- grid.points, 46
- guess.wavelength, 35
- hclust, 47
- hy.getOption (hy.getOptions), 35
- hy.getOptions, 35
- hy.setOptions, 39, 106
- hy.setOptions (hy.getOptions), 35
- hy.unittest, 37
- hy\_get\_option (Future-functions), 34
- hy\_set\_options (Future-functions), 34
- hyperSpec (hyperSpec-package), 5
- hyperSpec-class, 37

- hyperSpec-package, 5
- I, 114
- i2wl (wl2i), 117
- identify, 70
- image, 66
- import (read.txt.wide), 89
- initialize, 18, 38, 79, 80, 115
- initialize, hyperSpec-method
  - (initialize), 38
- invcm2ev (wlconv), 119
- invcm2freq (wlconv), 119
- invcm2nm (wlconv), 119
- invcm2raman (wlconv), 119
- is.na, hyperSpec-method (Summary), 110
- isample (sample, hyperSpec-method), 94
- isTRUE, 27, 110
- labels, 41
- labels, hyperSpec-method (labels<-), 40
- labels<-, 40
- labels<-, hyperSpec-method (labels<-), 40
- laser, 41
- legendright, 42
- length, 55, 97
- length, hyperSpec-method
  - (ncol, hyperSpec-method), 54
- levelplot, 46, 62–66, 112, 113
- levelplot, formula, hyperSpec-method
  - (plotmap), 62
- levelplot, hyperSpec, missing-method
  - (plotmap), 62
- lgamma, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- lines, 68, 70
- load, 89
- locator, 70, 100, 103
- loess, 104
- loess.control, 104
- log (Math2, hyperSpec-method), 49
- log, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- log1p, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- make.names, 35, 75
- makeraster, 44
- map.identify, 35, 46, 103
- map.identify (plotmap), 62
- map.sel.poly, 45, 64, 65, 103
- mark.dendrogram, 47
- markpeak, 48
- Math, 11, 14, 27, 50
- Math (Math2, hyperSpec-method), 49
- Math, hyperSpec-method
  - (Math2, hyperSpec-method), 49
- Math2 (Math2, hyperSpec-method), 49
- Math2, hyperSpec-method, 49
- matlab.dark.palette (matlab.palette), 50
- matlab.palette, 50
- matmult, 14
- matplot, 60
- max, hyperSpec-method (Summary), 110
- mean, 52
- mean, hyperSpec-method
  - (mean\_sd, numeric-method), 51
- mean\_pm\_sd (mean\_sd, numeric-method), 51
- mean\_pm\_sd, hyperSpec-method
  - (mean\_sd, numeric-method), 51
- mean\_pm\_sd, matrix-method
  - (mean\_sd, numeric-method), 51
- mean\_pm\_sd, numeric-method
  - (mean\_sd, numeric-method), 51
- mean\_sd (mean\_sd, numeric-method), 51
- mean\_sd, hyperSpec-method
  - (mean\_sd, numeric-method), 51
- mean\_sd, matrix-method
  - (mean\_sd, numeric-method), 51
- mean\_sd, numeric-method, 51
- melt, 16
- merge, 20, 54
- merge
  - (merge, hyperSpec, hyperSpec-method), 53
- merge(), 24
- merge, data.frame, hyperSpec-method
  - (merge, hyperSpec, hyperSpec-method), 53
- merge, hyperSpec, data.frame-method
  - (merge, hyperSpec, hyperSpec-method), 53
- merge, hyperSpec, hyperSpec-method, 53
- merge.data.frame, 53
- min, hyperSpec-method (Summary), 110
- ncol, 55
- ncol, hyperSpec-method, 54
- new, 39

- new (initialize), 38
- new,hyperSpec-method (initialize), 38
- nm2ev (wlconv), 119
- nm2freq (wlconv), 119
- nm2invcm (wlconv), 119
- nm2raman (wlconv), 119
- normalize.colrange (legendright), 42
- normalize.minmax (legendright), 42
- normalize.null (legendright), 42
- normalize.range (legendright), 42
- normalize01, 36, 56, 114, 116, 117
- normalize01,hyperSpec-method (normalize01), 56
- normalize01,matrix-method (normalize01), 56
- normalize01,numeric-method (normalize01), 56
- nrow, 55
- nrow,hyperSpec-method (ncol,hyperSpec-method), 54
- nwl (ncol,hyperSpec-method), 54
  
- Operators (Comparison), 25
- options, 69, 100, 105
- order, 57, 90
- orderwl, 53, 57, 118
  
- panel.identify, 64
- panel.levelplot.raster, 64
- panel.voronoi, 64, 65
- par, 68, 70, 101
- paracetamol, 58
- pearson.dist, 58
- plot, 60, 65, 68–70
- plot (plot-methods), 59
- plot,ANY,ANY-method (plot-methods), 59
- plot,hyperSpec,character-method (plot-methods), 59
- plot,hyperSpec,missing-method (plot-methods), 59
- plot-methods, 59
- plotc, 30, 59, 60, 61, 72
- plotmap, 30, 46, 59, 60, 62, 73
- plotmat, 60, 65
- plotmath, 39, 40
- plotspc, 30, 36, 59, 60, 67, 67, 70, 75, 100, 101, 103
- plotvoronoi, 46, 60
- plotvoronoi (plotmap), 62
  
- point.in.polygon, 46
- polygon, 69, 70
- pooled.cov, 93
- pooled.cov (cov,hyperSpec,missing-method), 28
- prcomp, 30
- princomp, 30
- print, 15
- print (as.character,hyperSpec-method), 14
- print,hyperSpec-method (as.character,hyperSpec-method), 14
- prod,hyperSpec-method (Summary), 110
  
- qmixlegend, 73
- qmixlegend (legendright), 42
- qmixtile, 73, 74
- qmixtile (legendright), 42
- qplotc, 71
- qplotmap, 72
- qplotmixmap, 73
- qplotspc, 36, 74
- quantile, 52
- quantile,hyperSpec-method (mean\_sd,numeric-method), 51
  
- R.matlab, 91
- rainbow, 50, 51
- raman2ev (wlconv), 119
- raman2freq (wlconv), 119
- raman2invcm (wlconv), 119
- raman2nm (wlconv), 119
- range,hyperSpec-method (Summary), 110
- rbind, 20, 36, 54, 75, 76
- rbind(), 24
- rbind.fill (rbind.fill.matrix), 75
- rbind.fill.matrix, 75
- rbind.hyperSpec (bind), 19
- rbind2, 20
- rbind2,hyperSpec,hyperSpec-method (bind), 19
- rbind2,hyperSpec,missing-method (bind), 19
  
- read.asc.PerkinElmer, 76
- read.cytomat, 77
- read.ENVI, 35, 78, 91
- read.ini, 81

- read.jdx, [82](#)
- read.mat.Cytospec (read.cytomat), [77](#)
- read.spc, [83](#), [85](#), [91](#)
- read.spc.Kaiser, [85](#)
- read.spc.KaiserLowHigh  
(read.spc.Kaiser), [85](#)
- read.spc.KaiserMap (read.spc.Kaiser), [85](#)
- read.spe, [86](#)
- read.table, [90](#), [91](#)
- read.txt.Horiba, [87](#)
- read.txt.long, [77](#)
- read.txt.long (read.txt.wide), [89](#)
- read.txt.Renishaw, [91](#)
- read.txt.Shimadzu, [88](#)
- read.txt.wide, [88](#), [89](#)
- read\_txt\_long (Future-functions), [34](#)
- read\_txt\_wide (Future-functions), [34](#)
- readLines, [81](#), [88](#)
- readMat, [91](#)
- rect, [48](#)
- rmmvnorm, [93](#)
- rmmvnorm, hyperSpec-method (rmmvnorm), [93](#)
- rmmvnorm, numeric, hyperSpec, array-method  
(rmmvnorm), [93](#)
- rmmvnorm, numeric, hyperSpec, matrix-method  
(rmmvnorm), [93](#)
- rmmvnorm, numeric, matrix, array-method  
(rmmvnorm), [93](#)
- rmmvnorm, numeric, matrix, matrix-method  
(rmmvnorm), [93](#)
- rmvnorm, [93](#)
- round, hyperSpec-method  
(Math2, hyperSpec-method), [49](#)
- rowMeans, [24](#)
- rowMeans, hyperSpec-method (colSums), [24](#)
- rownames, [31](#)
- rownames (dimnames, hyperSpec-method), [31](#)
- rownames, hyperSpec-method  
(dimnames, hyperSpec-method), [31](#)
- rownames<- (dimnames, hyperSpec-method),  
[31](#)
- rownames<- , hyperSpec-method  
(dimnames, hyperSpec-method), [31](#)
- rowSums, [24](#)
- rowSums, hyperSpec-method (colSums), [24](#)
- S4groupGeneric, [14](#), [27](#), [50](#)
- sample, [95](#)
- sample, data.frame-method  
(sample, hyperSpec-method), [94](#)
- sample, hyperSpec-method, [94](#)
- sample, matrix-method  
(sample, hyperSpec-method), [94](#)
- save, [89](#)
- scale, [95](#), [96](#)
- scale (scale, hyperSpec-method), [95](#)
- scale, hyperSpec-method, [95](#)
- scale-methods (scale, hyperSpec-method),  
[95](#)
- sd, [52](#)
- sel.poly, [46](#)
- sel.poly (map.sel.poly), [45](#)
- seq, [97](#)
- seq (seq.hyperSpec), [96](#)
- seq, hyperSpec-method (seq.hyperSpec), [96](#)
- seq.hyperSpec, [96](#)
- seq.int, [97](#)
- seq\_along, [97](#)
- show, [15](#)
- show (as.character, hyperSpec-method), [14](#)
- show, hyperSpec-method  
(as.character, hyperSpec-method),  
[14](#)
- sign, hyperSpec-method  
(Math2, hyperSpec-method), [49](#)
- signif, [115](#), [116](#)
- signif, hyperSpec-method  
(Math2, hyperSpec-method), [49](#)
- sin, hyperSpec-method  
(Math2, hyperSpec-method), [49](#)
- sinh, hyperSpec-method  
(Math2, hyperSpec-method), [49](#)
- smooth.spline, [105–107](#)
- spc.bin, [97](#), [116](#)
- spc.fit.poly, [99](#), [106](#)
- spc.fit.poly.below, [35](#), [106](#)
- spc.identify, [35](#), [64](#), [65](#), [69](#), [100](#)
- spc.label.default (spc.identify), [100](#)
- spc.label.wlonly (spc.identify), [100](#)
- spc.loess, [104](#), [107](#), [116](#)
- spc.NA.approx, [105](#)
- spc.NA.linapprox (spc.NA.approx), [105](#)
- spc.point.default (spc.identify), [100](#)
- spc.point.max (spc.identify), [100](#)
- spc.point.min (spc.identify), [100](#)
- spc.point.sqr (spc.identify), [100](#)

- spc.rubberband, [106](#)
- spc.smooth.spline, [107](#)
- spe.showcalpoints (read.spe), [86](#)
- split, [108](#), [108](#)
- split, ANY-method (split), [108](#)
- split, hyperSpec-method (split), [108](#)
- split-methods (split), [108](#)
- sqrt, hyperSpec-method  
(Math2, hyperSpec-method), [49](#)
- stack, [16](#)
- stacked.offsets, [68](#)
- stacked.offsets (plotspc), [67](#)
- subset, [109](#), [109](#)
- subset, hyperSpec-method (subset), [109](#)
- sum, hyperSpec-method (Summary), [110](#)
- Summary, [50](#), [110](#), [110](#)
- summary, [15](#)
- summary  
(as.character, hyperSpec-method),  
[14](#)
- Summary, hyperSpec-method (Summary), [110](#)
- summary, hyperSpec-method  
(as.character, hyperSpec-method),  
[14](#)
- sweep, [13](#), [27](#), [111](#), [111](#), [112](#)
- sweep, hyperSpec-method (sweep), [111](#)
- sweep-methods (sweep), [111](#)
  
- tan, hyperSpec-method  
(Math2, hyperSpec-method), [49](#)
- tanh, hyperSpec-method  
(Math2, hyperSpec-method), [49](#)
- tapply, [9](#)
- testthat, [37](#)
- text, [47](#), [48](#), [101](#), [102](#)
- textio, [81](#), [84](#)
- title, [69](#), [70](#)
- trellis.factor.key, [63](#), [65](#), [112](#)
- trigamma, hyperSpec-method  
(Math2, hyperSpec-method), [49](#)
- trunc, hyperSpec-method  
(Math2, hyperSpec-method), [49](#)
  
- unit, [64](#)
- unzip, [91](#)
  
- validObject, [22](#)
- validObject (chk.hy), [21](#)
  
- validObject, hyperSpec-method (chk.hy),  
[21](#)
- vanderMonde, [56](#), [113](#), [117](#)
- vanderMonde, hyperSpec-method  
(vanderMonde), [113](#)
  
- wc, [114](#)
- wl, [31](#), [115](#)
- wl.eval, [56](#), [114](#), [116](#)
- wl2i, [96](#), [97](#), [117](#), [121](#), [123](#)
- wl<- (wl), [115](#)
- wl\_convert\_units (Future-functions), [34](#)
- wlconv, [119](#)
- write.table, [90](#), [91](#)
- write.txt.long (read.txt.wide), [89](#)
- write.txt.wide (read.txt.wide), [89](#)
- writeMat, [91](#)
  
- xyplot, [61](#), [62](#)