Package 'geoSpectral'

July 3, 2017

Type Package

Title Classes and Methods for Working with Spectral Data with Space-Time Attributes

Version 0.17.4

Date 2017-03-22

Depends R (>= 2.10.0)

Imports methods, dplyr, spacetime, xts, maps, rgdal, leaflet, rbokeh, plotly, sp, stats

Author Servet Ahmet Cizmeli

Maintainer Servet Ahmet Cizmeli <ahmet@pranageo.com>

Description Provides S4 classes and data import, preprocessing, graphing, manipulation and export methods for geo-Spectral datasets (datasets with space/time/spectral dimensions). These type of data are frequently collected within earth observation projects (remote sensing, spectroscopy, bio-optical oceanography, mining, agricultural, atmospheric, environmental or similar branch of science).

License GPL

LazyLoad yes

Collate geoSpectral.R SpcHeader-Class.R SpcHeader-Methods.R Spectra-Class.R Spectra-Methods.R SpcList-Class.R SpcList-Methods.R Spectra-ImportFuns.R

RoxygenNote 6.0.1

Suggests testthat, xlsx

NeedsCompilation no

Repository CRAN

Date/Publication 2017-07-02 22:07:52 UTC

R topics documented:

`Arith,Spectra,Spectra-method`

Apply arithmetic operations on and between `Spectra` *objects.*

Description

Methods defining Arithmetic and Math operations between two `Spectra` objects e1 and e2 or one `Spectra` object e1 and a numeric value.

Usage

```
## S4 method for signature 'Spectra,Spectra'
Arith(e1, e2)

## S4 method for signature 'Spectra,numeric'
Arith(e1, e2)

## S4 method for signature 'Spectra'
Math(x)
```

Arguments

e1	spectra object
e2	spectra object or other
x	spectra object

Details

These methods allow performing arithmetic operations involving Spectra objects.

See Also

Arith

---

dim,Spectra-method *Dimensions of a* Spectra *object.*

Description

Gives number of dimension of a Spectra object

Usage

```
## S4 method for signature 'Spectra'
dim(x)
```

Arguments

x	A Spectra object

Value

Returns a numeric vector containing nrow and ncol of the Spectra object.

Examples

```
sp<-spc.example_spectra()
dim(sp)
```

`geoSpectral` *Classes and Methods for Working with Spectral Data with Space-Time Attributes*

Description

Provides S4 classes and data import, preprocessing, graphing, manipulation and export methods for geo-Spectral datasets (datasets with space/time/spectral dimensions). These type of data are frequently collected within earth observation projects (remote sensing, spectroscopy, bio-optical oceanography, mining, agricultural, atmospheric, environmental or similar branch of science).

Details

This package provides the following S4 classes:

- `Spectra` (stores spatial/temporal/spectral aspects of data)
- `SpcHeader` (stores metadata in an R list object)
- `SpcList` (makes a collection of Spectra objects in an R list)

as well as basic data access and manipulation methods for importing, acessing and subsetting, converting into R objects, analyzing, plotting, and exporting to other scientific data formats. Have a look at the constructor function by typing `?Spectra` to get started.

Author(s)

Servet Ahmet Cizmeli <ahmet@pranageo.com>

References

There is a tutorial for **geoSpectral** at the package's GitHub page: https://pranageo.com/geospectral/geospectral-tutorial/.

See Also

See also the packages **spacetime**, **rgdal**,'**sp**, **xts**

`head,Spectra-method` *Return the first or last part of a* `Spectra` *object*

Description

Return the first or last parts of a `Spectra` object

Usage

```
## S4 method for signature 'Spectra'
head(x, ...)
```

Arguments

x	a Spectra object
...	arguments to be passed to or from other methods

Value

Returns a matrix (Spectra data)

Examples

```
x <- spc.example_spectra()
head(x)
```

---

names,SpcList-method *names of* SpcList *object*

Description

Retrieve names of a SpcList object

Usage

```
## S4 method for signature 'SpcList'
names(x)
```

Arguments

x	A SpcList object

Value

Returns the coulmn names of an object of class SpcList as a charecter vector.

Examples

```
sp <- spc.example_spectra()
BL = spc.makeSpcList(sp,"CAST")
names(BL)
```

names,Spectra-method *The Names of a* Spectra *object*

Description

Retrieve the names of Spectra object

Usage

```
## S4 method for signature 'Spectra'
names(x)
```

Arguments

x	a Spectra object

Examples

```
x <- spc.example_spectra()
names(x)
```

ncol,Spectra-method *The Number of Columns of a Spectra object*

Description

nrow and ncol return the number of rows or columns of a Spectra object

Usage

```
## S4 method for signature 'Spectra'
ncol(x)
```

Arguments

x	A Spectra object

Examples

```
x <- spc.example_spectra()
ncol(x)  #501
nrow(x)  #26
```

nrow,Spectra-method *The Number of rows of a* Spectra *object*

Description

nrow and ncol return the number of rows or columns present in a Spectra object

Usage

```
## S4 method for signature 'Spectra'
nrow(x)
```

Arguments

x	a Spectra object

Examples

```
x <- spc.example_spectra()
ncol(x)  #501
nrow(x)  #26
```

rep,Spectra-method *Replicate rows of* Spectra *object*

Description

Operators

Usage

```
## S4 method for signature 'Spectra'
rep(x, times, ...)
```

Arguments

x	A Spectra object whose rows are to be replicated.
times	A integer vector giving the (non-negative) number of times to repeat each row. See help of rep.
...	further arguments to be passed to or from other methods. See help of rep.

Details

Replicates rows of x, making times copies of each row. Replicates Spectra, data, sp, time, endTime, InvalidIdx slots. Resets the SelectedIdx slot.

Value

A Spectra object

Examples

```
sp=spc.example_spectra()
dim(sp)
sp2 = rep(sp, 5)
dim(sp2)
```

---

show,SpcHeader-method *Show a SpcHeader object*

Description

Display a SpcHeader object

Usage

```
## S4 method for signature 'SpcHeader'
show(object)
```

Arguments

object	of class SpcHeader

See Also

show

Examples

```
x=spc.example_spectra()
show(x@header)
```

show,SpcList-method *Show a* SpcList *object*

Description

Display a SpcList object

Usage

```
## S4 method for signature 'SpcList'
show(object)
```

Arguments

object	a SpcList object

Value

show returns an invisible NULL

Examples

```
x <- spc.example_spectra()
BL = spc.makeSpcList(x,"CAST")
show(BL)
```

show,Spectra-method *Show a* Spectra *object*

Description

Display a Spectra object

Usage

```
## S4 method for signature 'Spectra'
show(object)
```

Arguments

object	a Spectra object

Value

show returns an invisible NULL

Examples

```
x <- spc.example_spectra()
show(x)
```

`sort,SpcList-method` *Sort a SpcList object*

Description

Applies the sort() method for `Spectra` class to every element of a `SpcList` object. All the `Spectra` objects within the `SpcList` object gets sorted according to the specified criteria.

Usage

```
## S4 method for signature 'SpcList'
sort(x, decreasing = FALSE, na.last = NA, which.col,
  ...)
```

Arguments

`x`	A `SpcList` object
`decreasing`	Logical. If TRUE, then the rows are sorted in decreasing order. Passed on to the sort.idx() function from the base package. Default is FALSE.
`na.last`	for controlling the treatment of NAs. Passed on to the sort.idx() function from the base package. Default is NA.
`which.col`	A character, defining the name of the column to be used in the sorting
`...`	arguments to be passed to or from methods. See help of `sort`.

Examples

```
sp <- spc.example_spectra()
#Create an SpcList object (one separate Spectra object for each unique STATION)
spL <- spc.makeSpcList(sp,"STATION")
#Sort all Spectra objects with respect to their rows using the CAST column
spL.s <- sort(spL,which.col="CAST",decreasing=TRUE)
lapply(spL.s, function(x) as.character(x[["CAST"]]))
```

sort,Spectra-method *Sort a Spectra object*

Description

Sort a Spectra object with respect to its rows with respect to values of one given column (specified by which.col). Sorting with respect to multiple columns is not implemented yet.

Usage

```
## S4 method for signature 'Spectra'
sort(x, decreasing = FALSE, na.last = NA, which.col,
  ...)
```

Arguments

x	A Spectra object
decreasing	Logical. If TRUE, then the rows are sorted in decreasing order. Passed on to the sort.idx() function from the base package. Default is FALSE.
na.last	for controlling the treatment of NAs. Passed on to the sort.idx() function from the base package. Default is NA.
which.col	A character, defining the name of the column to be used in the sorting
...	arguments to be passed to or from methods. See help of sort.

Examples

```
sp <- spc.example_spectra()
sp2 <- sort(sp, which.col="Offset")
sp2$Offset
sp2 <- sort(sp, which.col="CAST", decreasing=TRUE)
sp2$CAST
```

spc.bbox2lines *Constructs a rectangle with a* Spectra *object*

Description

Constructs a rectangle of sp::Lines using the bounding box of a Spectra object.

Usage

```
spc.bbox2lines(object)

## S4 method for signature 'Spatial'
spc.bbox2lines(object)

## S4 method for signature 'STI'
spc.bbox2lines(object)

## S4 method for signature 'Spectra'
spc.bbox2lines(object)
```

Arguments

object	spectra object t

Examples

```
sp=spc.example_spectra()
spc.bbox2lines(sp)
```

---

spc.cname.construct *Generating column names for a* Spectra *object*

Description

Function for a Spectra object that generates column names made of a combination of @shortName and @Wavelenght slots. If value is omitted, the @ShortName slot is used.

Usage

```
spc.cname.construct(object, value)

## S4 method for signature 'Spectra'
spc.cname.construct(object, value)
```

Arguments

object	A variable of class Spectra
value	A character object

Value

vector of characters

Examples

```
sp <- spc.example_spectra()
spc.cname.construct(sp)
spc.cname.construct(sp,"Newvar")
```

`spc.colMeans` *Computes the mean along the rows of a* `Spectra` *object*

Description

Computes the mean along the rows of a `Spectra` object. The method finds the measurement closest in time to the mean time and keeps the spatial/time attributes as well as Ancillary data table (@data) associated to that measurement as that of the mean spectra

Usage

```
spc.colMeans(object)

## S4 method for signature 'Spectra'
spc.colMeans(object)
```

Arguments

`object`	a `Spectra` object

Examples

```
sp=spc.example_spectra()
spc.colMeans(sp)
```

`spc.colnames` *Column names of* `Spectra` *object*

Description

Set or retrieve column names of a `Spectra` object

Set column names of a `Spectra` object

Usage

```
spc.colnames(x)

## S4 method for signature 'Spectra'
spc.colnames(x)

spc.colnames(x) <- value

## S4 replacement method for signature 'Spectra'
spc.colnames(x) <- value
```

Arguments

x	A Spectra object
value	character vector containing new column names to be assigned

Value

spc.colnames() returns the column names of an object of class Spectra as a character vector. spc.colnames()<- returns a Spectra object.

See Also

spc.cname.construct

Examples

```
x <- spc.example_spectra()
head(spc.colnames(x))
# or
spc.colnames(x) <- spc.cname.construct(x)
spc.colnames(x)
```

---

spc.data2header *Populate fields of header slot using data from data slot*

Description

Populates a field of @header with a column data from @data slot.

Usage

```
spc.data2header(object,dataname,headerfield,compress,...)

## S4 method for signature 'Spectra'
spc.data2header(object, dataname, headerfield,
  compress = FALSE, ...)
```

Arguments

object	A Spectra object.
dataname	A character object specifying the name of @data column to be used.
headerfield	A character object specifying the name of the @header field to be changed
compress	logical. Whether or not to compress data put into the header. See the description section.
...	arguments to be passed to or from other methods

Details

This function extracts data from a column of the @data slot (specified by dataname) and creates a new @header field with it. Ifa header field is not provided, the name of the new header field will be the same as dataname.

The name of the new header field can be overwritten by providing header field. If all the incoming data rows (dataname) are the same, information put into the header can be compressed by selecting compress=TRUE (default is FALSE). This would take only the first element from the @data column.

Value

object of class Spectra

Examples

```
sp=spc.example_spectra()
sp=spc.data2header(sp,"CAST")
sp@header
sp=spc.data2header(sp,"CAST","ProjectCast")
sp@header
sp$CAST=rep(33, nrow(sp))
sp=spc.data2header(sp,"CAST","ProjectCast", compress=TRUE)
sp@header
```

---

spc.data2header,list-method

Populate fields of header slot using data from data slot

Description

Populates a field of @header with a column data from @data slot.

Usage

```
## S4 method for signature 'list'
spc.data2header(object, dataname, headerfield,
  compress = TRUE, ...)
```

Arguments

object	SpcList object
dataname	A character object specifying the name of @data column to be used
headerfield	A character object specifying the name of the @header field to be changed
compress	TRUE or FALSE
...	arguments to be passed to or from methods. See help of spc.data2header.

Details

This function extracts data from a column of the @data slot (specified by dataname) and creates a new @header field with it. If headerfield is not provided, the name of the new header field will be the same as dataname. The name of the new header field can be overwritten by providing headerfield. If all the incoming data rows (dataname) are the same, information put into the header can be compressed by selecting compress=TRUE (default is FALSE). This would take only the first element from the @data column.

Value

object of class SpcList

Examples

```
sp=spc.example_spectra()
BL=spc.makeSpcList(sp,"CAST")
BL[[1]]@header
 BL[[1]]=spc.data2header(BL[[1]],"CAST","ProjectCast")
BL[[1]]@header
BL[[1]]$CAST=rep(33, nrow( BL[[1]]))
BL[[1]]=spc.data2header(BL[[1]],"CAST","ProjectCast", compress=TRUE)
BL[[1]]@header
```

---

spc.example_spectra *Create example of Spectral object*

Description

Example of Spectral object is created by the function

Usage

```
spc.example_spectra()
```

Examples

```
sp = spc.example_spectra()
class(sp)
show(sp)
```

spc.export.text *Exporting into text format*

Description

Save the Spectra and SpcHeader objects on disk in text format and read back in.

Usage

```
spc.export.text(input, filename, sep = ";", append = FALSE,
  writeheader = TRUE, ...)

## S4 method for signature 'Spectra'
spc.export.text(input, filename, sep = ";",
  append = FALSE, writeheader = TRUE, ...)

## S4 method for signature 'SpcHeader'
spc.export.text(input, filename, sep = ";",
  append = FALSE, writeheader = TRUE, ...)

spc.import.text(filename, sep = ";", ...)
```

Arguments

input	A Spectra object
filename	Name of the output text file
sep	character. the field separator string
append	logical. Only relevant if file is a character string. Default is TRUE
writeheader	either a logical value indicating whether the header names are to be written
...	arguments to be passed to or from other methods

See Also

spc.import.text

Examples

```
x=spc.example_spectra()
spc.export.text(x,filename="anap.txt")
aa=spc.import.text("anap.txt")
dev.new()
spc.plot(aa)

#Export the SpcHeader object
spc.export.text(x@header,filename="anap_header.txt")
hdr=spc.import.text("anap_header.txt")
class(hdr)
```

`spc.export.xlsx` *Exports a* `Spectra` *object into Excel format.*

Description

Exports a `Spectra` object into Excel format.

Usage

```
spc.export.xlsx(input, filename, sheetName, writeheader = TRUE, append = F,
  sep = ";", ...)

## S4 method for signature 'Spectra'
spc.export.xlsx(input, filename, sheetName,
  writeheader = TRUE, append = F, sep = ";", ...)
```

Arguments

`input`	A `Spectra` object
`filename`	Name of the output xlsx file
`sheetName`	The `Spectra` object to be output.
`writeheader`	A boolean, indicating whether or not the metadata (contents of the slot `header`) is to be included in the excel file. Default : TRUE
`append`	A boolean, indicating whether or not to append the contents of the `Spectra` object into the existing file. Default : FALSE (overwrites the existing Excel file if it exists.)
`sep`	Not used.
`...`	Not used.

Details

`spc.export.xlsx()` calls functions from package `xlsx` to write the contents of a `Spectra` object into an Excel file. For this function to work, make sure the package `xlsx` is installed.

Value

None. Simply creates an Excel file on disk.

Examples

```
## Not run:
  sp=spc.example_spectra()
  if("xlsx" %in% installed.packages())
     spc.export.xlsx(sp,"test.xlsx")

## End(Not run)
```

spc.getheader *Extract a field of the @header slot of a* Spectra *object*

Description

Extracts the value of a field in the header slot of Spectra object

Usage

```
spc.getheader(object,name)

## S4 method for signature 'Spectra'
spc.getheader(object, name)
```

Arguments

object	A Spectra object
name	of the header field to be extracted

See Also

spc.setheader<-

Examples

```
sp=spc.example_spectra()
sp@header
spc.getheader(sp,"Latitude")
```

spc.getheader,list-method *Extract a field of the @header slot of a* SpcList *object*

Description

Extracts the value of a field in the header slot of SpcList object

Usage

```
## S4 method for signature 'list'
spc.getheader(object, name)
```

Arguments

object	A SpcList object
name	of the header field to be extracted

Examples

```
sp=spc.example_spectra()
BL = spc.makeSpcList(sp,"CAST")
BL[[1]]@header
spc.getheader(BL,"CAST")
```

---

spc.getinvalid.idx *Get index of* Spectra *rows marked as invalid*

Description

Extract the row indexes stored as invalid

Usage

```
spc.getinvalid.idx(object)

## S4 method for signature 'Spectra'
spc.getinvalid.idx(object)
```

Arguments

object	A Spectra object

Value

Logical vector

Examples

```
sp= spc.example_spectra()
spc.getinvalid.idx(sp) #No invalid rows
```

---

spc.getselected.idx *Extract index inside of a* Spectra *object*

Description

Extracts index of rows marked as selected

Usage

```
spc.getselected.idx(object)

## S4 method for signature 'Spectra'
spc.getselected.idx(object)
```

Arguments

object	A Spectra object

Value

Spectra object

See Also

spc.setselected.idx<-

Examples

```
x <- spc.example_spectra()
idx=rep(FALSE,nrow(x));
idx[1:5]=TRUE
spc.setselected.idx(x)<-idx
spc.getselected.idx(x)
```

---

spc.getwavelengths *Extract wave lenghts of a* Spectra *object*

Description

Get wave lenghts inside of a Spectra object

Usage

```
spc.getwavelengths(object)

## S4 method for signature 'Spectra'
spc.getwavelengths(object)
```

Arguments

object	A Spectra object

Value

numeric vector of wave lengths

See Also

spc.setwavelengths<-

Examples

```
x <- spc.example_spectra()
spc.getwavelengths(x)
```

---

spc.header.infos *Getting as input the* Spectra *heade*

Description

This internal function takes as input the Spectral header as a list and converts its elements to numbers (when possible) evals its elements in case the text contains some R code

Usage

```
spc.header.infos(header)
```

Arguments

header	A Spectra header

Examples

```
sp=spc.example_spectra()
spc.header.infos(sp@header)
```

---

spc.header2data *Copy header data into the @data slot*

Description

Get the header metadata and place it inside the @data slot

Usage

```
spc.header2data(object, headerfield, dataname, compress, ...)

## S4 method for signature 'Spectra'
spc.header2data(object, headerfield, dataname,
  compress = TRUE, ...)

## S4 method for signature 'list'
spc.header2data(object, headerfield, dataname,
  compress = TRUE, ...)
```

```
## S4 method for signature 'SpcList'
spc.header2data(object, headerfield, dataname,
  compress = TRUE, ...)
```

Arguments

object	A Spectra object
headerfield	character. Field name of the header to be copied.
dataname	character. Column name of @data slot to copy the incoming data.
compress	logical. Whether or not to compress data put into the header. See help of spc.data2header.
...	arguments to be passed to or from other methods

Details

If header element has length >1, its type is checked. If it is "character", its elements will be pasted using paste(...,collapse="|"). If it is another type, only the first element will be taken. For list and SpcList objects, the same procedure is repeated for all elements of the list containing Spectra objects. If dataname is missing, then it will be taken equal to headerfield.

Value

object of class Spectra or SpcList

Examples

```
sp <- spc.example_spectra()
sp <- spc.updateheader(sp,"Zone", "ZoneA")
sp <- spc.header2data(sp, "Zone")
sp$Zone
```

---

spc.interp.spectral *Interpolate spectral values*

Description

Estimate spectral data at a new set of wavelengths through interpolation using approx().

Usage

```
spc.interp.spectral(source1,target_lbd,show.plot, ...)

## S4 method for signature 'Spectra'
spc.interp.spectral(source1, target_lbd,
  show.plot = FALSE)
```

Arguments

source1	A Spectra object
target_lbd	numeric vector giving desired wavelengths
show.plot	logical TRUE if a graphical representation is required
...	further arguments to pass on to approx().

Examples

```
sp=spc.example_spectra()
lbd = as.numeric(c(412,440,490,555,670))
sp2 = spc.interp.spectral(sp[,lbd],c(430,450,500))
spc.plot.overlay(SpcList(list(sp,sp2)))

#Quick Plot only the first row
spc.interp.spectral(sp[,lbd],c(430,450,500),show.plot=TRUE)
```

---

spc.invalid.detect *Determinate invalid rows of a* Spectra *object*

Description

Determine invalid rows (records) of a Spectra SpcList object

Determine invalid records inside a SpcList object

Usage

```
spc.invalid.detect(source1)

## S4 method for signature 'Spectra'
spc.invalid.detect(source1)

## S4 method for signature 'list'
spc.invalid.detect(source1)
```

Arguments

source1	A Spectra object

Value

logical. TRUE for invalid rows

Examples

```
sp=spc.example_spectra()
nrow(sp)
invalid=spc.invalid.detect(sp)
show(invalid); length(invalid)

BL = spc.makeSpcList(sp,"CAST")
invalid=spc.invalid.detect(BL)
show(invalid)
```

`spc.lapply` *Apply a function over a Spclist*

Description

lapply returns a list of the same length as X, each element of which is the result of applying FUN to the corresponding element of X.

Usage

```
spc.lapply(X, FUN, ...)

## S4 method for signature 'SpcList'
spc.lapply(X, FUN, ...)
```

Arguments

`X`	A `SpcList` object .
`FUN`	function to be applied to each element of X.
`...`	optional arguments to FUN.

Value

list or SpcList object.

Examples

```
sp=spc.example_spectra()
BL=spc.makeSpcList(sp,"CAST")
#Counts rows (returns a list object)
spc.lapply(BL,function(x) {nrow(x)})
#Perform arithmetic operations on all Spectra elements. Returns a SpcList object.
spc.lapply(BL,function(x) {x^2+1})
```

spc.lines *Add spectra to an existing plot*

Description

Adds spectra to an existing plot created by spc.plot() using lines()

Usage

```
spc.lines(x,...)

## S4 method for signature 'Spectra'
spc.lines(x, ...)
```

Arguments

x	An object of class Spectra
...	Additional input arguments to be passed to lines()

See Also

spc.plot

Examples

```
sp = spc.example_spectra()
spc.plot(sp[2,])
spc.lines(sp[3,],col="red")
```

spc.make.stindex *Create a spatio-temporal index based on a list of* Spectra *objects*

Description

Given a list of Spectra objects, this function creates a STIDF object summarizing the spatial and temporal variability of the input dataset. Upon request, it also includes data columns.

Usage

```
spc.make.stindex(input, what2include = "", rowSimplify = "none",
  includeTIME = FALSE, includeLATLON = FALSE)
```

Arguments

input	An object of class spectra
what2include	A character variable giving the data columns to be included in the output
rowSimplify	Either of "none", "spc.colMeans","firstRow" or "lastRow". Default is "none"
includeTIME	Logical. Whether of not to include TIME data in the output STIDF object. Default is FALSE.
includeLATLON	Logical. Whether of not to include LAT&LON data in the output STIDF object. Default is FALSE.

Details

This function accepts a list of Spectra objects and outputs one STIDF object summarizing spatial and temporal variation of the input dataset.

If rowSimplify="none", length of the output object will be equal to the sum of all rows of all elements of the input list object.

If rowSimplify="spc.colMeans", length of the output object will be equal to the number of rows of the input list object. This option returns the measurement nearest to the average time of each element of the input list.

firstRow and lastRow : length of the output object equals the number of rows of the input list object. These two options return the first and last measurements of the input list element

Value

An object of class STIDF. Each row of the output object has a space and time characteristics depending of the input argument rowSimplify.

See Also

spc.makeSpcList

Examples

```
sp = spc.example_spectra()
BL = spc.makeSpcList(sp,"STATION")
stidx = spc.make.stindex(BL)
dim(stidx)
stidx = spc.make.stindex(BL, what2include = "CAST")
head(stidx@data)
stidx = spc.make.stindex(BL, rowSimplify="spc.colMeans")
dim(stidx)
```

spc.makeSpcList *Conversion from* Spectra *to* Spclist

Description

Conversion from Spectra to Spclist using a data field

Usage

```
spc.makeSpcList(myobj, name)
```

Arguments

myobj	a Spectra object
name	name of station of a Spectra object

Examples

```
sp <- spc.example_spectra()
BL = spc.makeSpcList(sp,"CAST")
show(BL)
```

spc.plot *Plotting* Spectra *object*

Description

Generating plot of the intensity of a measurement inside a Spectra object with respect to the wavelength.

Usage

```
spc.plot(x, Y, maxSp, lab_cex,xlab,ylab,type,pch,lwd,cex,...)

## S4 method for signature 'Spectra'
spc.plot(x, Y, maxSp, lab_cex, xlab, ylab, type = "l",
  pch = 19, lwd = 2, cex = 0.3, ...)
```

Arguments

x	and Y a Spectra data
Y	fskjldsk
maxSp	maximum number of Spectra to plot
lab_cex	vector of character expansion sizes, used cyclically
xlab	title for x axix, as in plot().
ylab	title for y axis, as in plot().
type	character string (length 1 vector) or vector of 1-character strings indicating the type of plot for each column of y. See help of matplot() or plot().
pch	character string or vector of 1-characters or integers for plotting characters. See help of par.
lwd	vector of line widths. See help of par.
cex	A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default. See help of par.
...	any further arguments to be passed to matplot

See Also

spc.lines, par

Examples

```
x <- spc.example_spectra()
spc.plot(x)
```

---

spc.plot.depth *Plotting* Spectra *object*

Description

Generating plot of the contents of a Spectra object with respect to depth

Usage

```
spc.plot.depth(object, ...)

## S4 method for signature 'Spectra'
spc.plot.depth(object, X, maxSp = 10, lab_cex, title,
  add = FALSE, xlab = NULL, ylab = NULL, ylim = NULL, xlim = NULL,
  lwd = 2, ...)
```

Arguments

object	a Spectra data.
...	any further arguments of plot
X	character. Column names of the a Spectra object to be plotted.
maxSp	numeric. Maximum number of Spectra to plot.
lab_cex	vector of character expansion sizes, used cyclically.
title	a chracter string, title for plot
add	logical. If TRUE, plots are added to current one,
xlab, ylab	titles for x and y axes, as in plot.
ylim, xlim	ranges of x and y axes, as in plot.
lwd	numeric vector of line widths

See Also

spc.plot

Examples

```
x <- spc.example_spectra()
spc.plot.depth(x)
```

---

spc.plot.depth.overlay

Plotting SpcList *object*

Description

Generating plot of the contents of a SpcList object overlay with respect to depth

Usage

```
spc.plot.depth.overlay(object, X, lab_cex, ...)

## S4 method for signature 'SpcList'
spc.plot.depth.overlay(object, X, lab_cex, ...)
```

Arguments

object	a SpcList data
X	column name or index
lab_cex	vector of character expansion sizes, used cyclically
...	any further arguments of plot

Examples

```
sp <- spc.example_spectra()
BL = spc.makeSpcList(sp,"CAST")
spc.plot.depth.overlay(BL, "anap_555")
```

`spc.plot.depth.plotly` *Display a Spectra object*

Description

Plot a `Spectra` object with respect to depth

Usage

```
spc.plot.depth.plotly(sp, column, plot.max = 10, showlegend = FALSE,
  hoverinfo = "name", title = sp@LongName)

## S4 method for signature 'Spectra'
spc.plot.depth.plotly(sp, column, plot.max = 10,
  showlegend = FALSE, hoverinfo = "name", title = sp@LongName)
```

Arguments

`sp`	A `Spectra` object
`column`	Number or name , default value is 10 if a number or name has not been entered
`plot.max`	numeric value for a maximum number of data in plot
`showlegend`	logical, to display legend or not, default is FALSE
`hoverinfo`	a chracter, info about `Spectra` object to be used in hover box
`title`	a chracter string, title for plot

Examples

```
sp = spc.example_spectra()
BL = spc.makeSpcList(sp,"CAST")
p1<-spc.plot.depth.plotly(BL[[5]])
#p1<-layout(p1,title=paste("CAST =", BL[[5]]$CAST[1]))
p2<-spc.plot.depth.plotly(BL[[4]])
#p2<-plotly::layout(p2,title=paste("CAST =", BL[[4]]$CAST[1]))
p <- plotly::subplot(p1, p2,  margin = 0.05, shareY=TRUE,shareX=TRUE,titleX=TRUE,titleY=TRUE)
p <- plotly::layout(p, showlegend = TRUE,
annotations = list(
list(x = 0.2 , y = 1.05, text = BL[[5]]$CAST[1], showarrow = FALSE, xref='paper', yref='paper'),
list(x = 0.8 , y = 1.05, text = BL[[4]]$CAST[1], showarrow = FALSE, xref='paper', yref='paper')))
p
```

spc.plot.grid *Plotting* SpcList *object in a grid*

Description

Generating plot of the contents of a SpcList object in a grid

Usage

```
spc.plot.grid(x,FUN, nnrow, nncol, mar,oma, lab_cex, ...)

## S4 method for signature 'SpcList'
spc.plot.grid(x, FUN, nnrow, nncol, mar = c(4, 4.5, 1,
  0.5), oma = c(0, 0, 0, 0), lab_cex, ...)
```

Arguments

x	a SpcList data
FUN	a character string giving the name of the ploting function to be used. Can be either of "spc.plot"
nnrow	number of rows for the grid to be produced
nncol	number of columns for the grid to be produced
mar	A numeric vector of length 4, which sets the margin sizes in the following order: bottom, left, top, and right. The default is c(4,4.5,1,0.5)
oma	oma the "outer margin area" around a figure or figures. The usage of mar and oma is shown when plotting a single figure,
lab_cex	vector of character expansion sizes, used cyclically
...	any further arguments of plot

Examples

```
sp <- spc.example_spectra()
BL = spc.makeSpcList(sp,"CAST")
spc.plot.grid(BL,"spc.plot",3,2)
```

spc.plot.map.leaflet *Display a Spectra object*

Description

Create a point map with leaflet engine using Spectra rows

Usage

```
spc.plot.map.leaflet(sp, hover_field = "row", color = "#FF0000",
  opacity = 1, weight = 5)

## S4 method for signature 'Spectra'
spc.plot.map.leaflet(sp, hover_field = "row",
  color = "#FF0000", opacity = 1, weight = 5)
```

Arguments

sp	Spectra object
hover_field	A character or vector of strings giving column names of Spectra object. This information will be displayed when hovering over the glyph
color	Determine color of points
opacity	The opacity transparency of the glyph between 0 (transparent) and 1 (opaque)
weight	Stroke width in pixels

Examples

```
sp=spc.example_spectra()
spc.plot.map.leaflet(sp)
```

spc.plot.map.plotly *Display a Spectra object*

Description

Create a point map with ploty engine using Spectra rows

Usage

```
spc.plot.map.plotly(sp, hover_field = "row", color = "#FF0000",
  opacity = 1)

## S4 method for signature 'Spectra'
spc.plot.map.plotly(sp, hover_field = "row",
  color = "#FF0000", opacity = 1)
```

Arguments

sp	A Spectra object
hover_field	A character, column names of sp object to be used in hover box
color	Determine color of points
opacity	The opacity transparency of the glyph between 0 (transparent) and 1 (opaque)

Examples

```
sp <- spc.example_spectra()
spc.plot.map.plotly(sp)
```

---

spc.plot.map.rbokeh *Display a Spectra object*

Description

Create a point map with rbokeh engine using Spectra rows

Usage

```
spc.plot.map.rbokeh(sp, glyph = 2, color = "#FF0000", legend = NULL,
  hover = "row", opacity = 1)

## S4 method for signature 'Spectra'
spc.plot.map.rbokeh(sp, glyph = 2, color = "#FF0000",
  legend = NULL, hover = "row", opacity = 1)
```

Arguments

sp	Spectra object
glyph	Value(s) or field name of the glyph to use point_types
color	Determine color of points
legend	not implemented yet
hover	String or vector of strings giving column names of Spectra object. This information will be displayed when hovering over the glyph
opacity	The opacity transparency of the glyph between 0 (transparent) and 1 (opaque)

Examples

```
## Not run:
  sp=spc.example_spectra()
  spc.plot.map.rbokeh(sp, hover = "Snap")
  spc.plot.map.rbokeh(sp)

## End(Not run)
```

`spc.plot.overlay` *Plotting multiple* `Spectra` *objects inside a* `SpcList`

Description

This function overlays spectra plots of several `Spectra` objects inside a `SpcList` object. The first element of the input `SpcList` object is plotted with spc.plot() while remaining elements are overlaid with spc.lines().

Usage

```
spc.plot.overlay(object,lab_cex,leg_idx, type, lty,lwd, col, ...)

## S4 method for signature 'SpcList'
spc.plot.overlay(object, lab_cex = 1, leg_idx = TRUE,
  type = "l", lty = 1, lwd = 1, col, ...)
```

Arguments

`object`	A `SpcList` data
`lab_cex`	vector of character expansion sizes, used cyclically
`leg_idx`	logical If it is of length 1, it determines whether or not to display the legend. If length(leg_idx) is bigger than 1, then its lengths has to equal length(object). Default is TRUE.
`type`	character string (length 1 vector) or vector of 1-character strings indicating the type of plot for each column of y,
`lty`	vector of line types. See par().
`lwd`	numeric. Vector of line widths. See par().
`col`	A specification for the default plotting color. See par().
`...`	any further arguments to the plotting function matplot() or spc.plot()

Examples

```
sp <- spc.example_spectra()
BL = spc.makeSpcList(sp,"CAST")
spc.plot.overlay(BL)
spc.plot.overlay(BL, xlim=c(400,500),ylim=c(0,0.2),lwd=2)
spc.plot.overlay(BL, col=c("red"), leg_idx=FALSE, lty=2)
spc.plot.overlay(BL, col=c("red","blue","green","yellow","cyan","black"))
```

spc.plot.plotly *Plot a Spectra object data*

Description

Plot a Spectra object with plotly engine

Usage

```
spc.plot.plotly(sp, plot.max = 10, showlegend = FALSE,
  legend_field = "row", hoverinfo = "title", title = sp@LongName)

## S4 method for signature 'Spectra'
spc.plot.plotly(sp, plot.max = 10, showlegend = FALSE,
  legend_field = "row", hoverinfo = "title", title = sp@LongName)
```

Arguments

sp	A Spectra object
plot.max	numeric value for a maximum number of data in plot. Default is 10.
showlegend	logical, to display legend or not, default is FALSE
legend_field	character. Gives the name of the column to be used in the legend.
hoverinfo	a chracter, info about Spectra object to be used in hover box.
title	a chracter string, title for plot.

Examples

```
sp = spc.example_spectra()
spc.plot.plotly(sp)
spc.plot.plotly(sp,legend_field = "Spectra")
spc.plot.plotly(sp,legend_field = "CAST")
spc.plot.plotly(sp,legend_field = "NISKIN")
spc.plot.plotly(sp,legend_field = "STATION")
spc.plot.plotly(sp,legend_field = "anap_440")
```

spc.plot.time *Plotting* Spectra *object*

Description

Generating plot of the contents of a Spectra object with respect to time. If xdata is 'time', data is plotted with respect to the 'TIME' column. If xdata is 'observations', data is plotted with respect to an integer index equal to 1:nrow(object).

Usage

```
spc.plot.time(object, ...)

## S4 method for signature 'Spectra'
spc.plot.time(object, Y, maxSp = 50, xdata = "time",
  lab_cex, lwd = 2, ...)
```

Arguments

object	A Spectra object.
...	any further arguments of plot
Y	character. Name of the columns of the Spectra object to be plotted.
maxSp	numeric. Maximum number of Spectra to plot.
xdata	character. Type of time-series data. Can be 'time' or 'observations'.
lab_cex	vector of character expansion sizes, used cyclically.
lwd	vector of line widths

See Also

spc.plot.depth

Examples

```
x <- spc.example_spectra()
spc.plot.time(x)
```

---

spc.plot.time.plotly *Plot a Spectra object data with respect to time*

Description

Plot a Spectra object with respect to time

Usage

```
spc.plot.time.plotly(sp, column, plot.max = 10, showlegend = FALSE,
  hoverinfo = "name", title = sp@LongName)

## S4 method for signature 'Spectra'
spc.plot.time.plotly(sp, column, plot.max = 10,
  showlegend = FALSE, hoverinfo = "name", title = sp@LongName)
```

Arguments

sp	A Spectra object
column	Number or name , defoult value is 10 if a number or name has not been entered
plot.max	numeric value for a maximum number of data in plot
showlegend	logical, to display legend or not, default is FALSE
hoverinfo	a chracter, info about Spectra object to be used in hover box
title	a chracter string, title for plot

Examples

```
## Not run:
sp = spc.example_spectra()
spc.plot.time.plotly(sp)
spc.plot.time.plotly(sp, plot.max = 3)
spc.plot.time.plotly(sp, c("anap_450","anap_550","anap_650"))

## End(Not run)
```

---

spc.rbind *Combine* Spectra *Objects by Rows*

Description

Take a Spectra objects and combine by rows

Usage

```
spc.rbind(...)

## S4 method for signature 'Spectra'
spc.rbind(..., compressHeader = TRUE)
```

Arguments

...	Spectra object
compressHeader	Compress the header (make multiple all-equal header elements as ONE, default value is TRUE

Value

Spectra object

Examples

```
x <- spc.example_spectra()
nrow(x)  #[1] 26
x2=spc.rbind(x,x)
nrow(x2)  #[1] 52
```

---

spc.rbind,STIDF-method *Combine* STIDF *objects by Rows*

Description

Take a STIDF objects and combine by rows

Usage

```
## S4 method for signature 'STIDF'
spc.rbind(...)
```

Arguments

...	STIDF object

Examples

```
x <- spc.example_spectra()
nrow(x)  #[1] 26
x2 <- spc.rbind(as(x, "STIDF"),as(x, "STIDF"))
nrow(x2)  #[1] 52
```

---

spc.Read_ASD *Read the ASD Spectra from text file*

Description

Imports ASD spectra from text files prepared by the software provided by ASD inc. This function imports only one spectra per file.

Usage

```
spc.Read_ASD(filename)
```

Arguments

filename	A string name of the input text file containing the raw ASD data.

Value

Returns an object of class Spectra.

Examples

```
filename = file.path(system.file(package = "geoSpectral"),"test_data", "106.064.txt")
L = spc.Read_ASD(filename)
class(L)
spc.plot.plotly(L)
```

---

spc.Read_NOMAD_v2 *Read the NOMAD v2 bio-optical database*

Description

Imports the NOMAD v2 database of the SeaBASS project. More information about this dataset can be found at https://seabass.gsfc.nasa.gov/wiki/NOMAD

Usage

```
spc.Read_NOMAD_v2(skip.all.na.rows = TRUE)
```

Arguments

skip.all.na.rows	logical whether or not eliminate records where all channels are NAs

Value

Returns an object of class data.frame.

Examples

```
nomad = spc.Read_NOMAD_v2()
class(nomad[[1]])
spc.plot.plotly(nomad[[4]], plot.max=15)
```

spc.select *Selecting rows of a* Spectra *object with the mouse*

Description

This function allows the selection of Spectra rows that is drawn with spc.plot or spc.lines. Selected lines will be colored red. Pressing the escape button will end the selection process and return selecion results.

Usage

```
spc.select(object)

## S4 method for signature 'Spectra'
spc.select(object)
```

Arguments

object	A Spectra object

Value

logical Row indexes, TRUE for selected data rows.

See Also

spc.plot spc.lines

Examples

```
sp <- spc.example_spectra()
spc.plot(sp)
spc.setselected.idx(sp)<-spc.select(sp)
```

spc.setheader<- *Set a field of the @header slot of a* Spectra *object*

Description

Function sets or changes the value of a field in the header slot of Spectra object

Usage

```
spc.setheader(object) <- value

## S4 replacement method for signature 'Spectra'
spc.setheader(object) <- value
```

Arguments

object	A Spectra object
value	Object of class SpcHeader

See Also

spc.getheader

Examples

```
sp=spc.example_spectra()
a=new("SpcHeader") # create new SpcHeader class
a$Longitude=123
spc.setheader(sp) <- a
sp@header
```

---

spc.setheader<-,list-method

Set a field of the @header slot of a SpcList *object*

Description

Function sets or changes the value of a field in the header slot of SpcList object.

Usage

```
## S4 replacement method for signature 'list'
spc.setheader(object) <- value
```

Arguments

object	A SpcList object.
value	Object of class SpcList.
...	arguments to be passed to or from other methods.

Examples

```
sp=spc.example_spectra()
BL=spc.makeSpcList(sp,"CAST")
a=new("SpcHeader") # create new SpcHeader class
a$Longitude=123
spc.setheader(BL[[1]]) <- a
h=spc.getheader(BL[[1]])
h
```

spc.setinvalid.idx<- *Set rows of* Spectra *as invalid*

Description

Stores the row indexes to be stored as invalid.

Usage

```
spc.setinvalid.idx(object) <- value

## S4 replacement method for signature 'Spectra'
spc.setinvalid.idx(object) <- value
```

Arguments

object	A Spectra object
value	Logical vector

See Also

spc.setselected.idx<-

Examples

```
sp = spc.example_spectra()
spc.getinvalid.idx(sp) #No invalid rows
vld = rep(TRUE,26)
vld[1:5]<-FALSE
spc.setinvalid.idx(sp)<-vld #Mark the first 5 rows as invalid
spc.getinvalid.idx(sp)
```

spc.setselected.idx<- *Set index to a* Spectra *object*

Description

Set or change selected row index of a Spectra object

Usage

```
spc.setselected.idx(object) <- value

## S4 replacement method for signature 'Spectra'
spc.setselected.idx(object) <- value
```

Arguments

object	A Spectra object
value	index for a Spectra object

Value

A Spectra object

See Also

spc.getselected.idx

Examples

```
x <- spc.example_spectra()
idx=rep(FALSE,nrow(x));
idx[1:5]=TRUE
spc.setselected.idx(x)<-idx
spc.plot(x)
```

---

spc.setwavelengths<- *Setting wavelengths in a* Spectra *object*

Description

Function to change or set wavelengths inside of a Spectra object

Usage

```
spc.setwavelengths(object) <- value

## S4 replacement method for signature 'Spectra'
spc.setwavelengths(object) <- value
```

Arguments

object	A Spectra object
value	Numeric

See Also

spc.getwavelengths

Examples

```
x <- spc.example_spectra()
show(x)
spc.setwavelengths(x) <- 300:800
show(x)
```

spc.STI.stdistance *Report the space and time distance of each row of an STI-inherited object*

Description

Function that reports the space and time distance of each row of the STI-inherited object `searched` to the corresponding row of the STI-inherited object `master`

Usage

```
spc.STI.stdistance(master, searched, report = F)
```

Arguments

`master`	An STI-inherited object
`searched`	An STI-inherited object
`report`	Logical. Default value is FALSE

Details

Reports the space and time distance of each row of the STI-inherited object `searched` to the corresponding row of the STI-inherited object `master`.

Value

Outputs a data.frame, with two columns : time2master ("difftime", in seconds) and distance2master ("numeric", in meters)

spc.timeMatch *Match two time sequences*

Description

Match two time sequences for a `Spectra` object, where each can be intervals or instances.

Usage

```
spc.timeMatch(master,searched,returnList,method,limits,report)
```

Arguments

master	ordered sequence of variable of class Spectra
searched	A variable of class Spectrawhich is searched
returnList	Boolean; should a list be returned with all matches (TRUE), or a vector with single matches (FALSE)?
method	Method used in time-based matching. See the details section.
limits	the interval limits
report	return character string which has information about searching results, default is False

Details

spc.timeMatch is similar to spacetime::timeMatch(), only adding some more matching methods. When method is "over", the same technique used by spacetime::timeMatch() is used. Useful when matched timestamps of both master and searched are exactly equal. When method is "nearest", the nearest measurement will be found, matching only one data for ALL elements of master. When method is "within", measurements that are within the interval limits=c(upper,lower) (in seconds) will be found.

Examples

```
#Read the Nomad database inside a SpcList object.
dat = SpcList(spc.Read_NOMAD_v2())

#Different list elements containt different parameters
names(dat)

#We would like to find elements of Es that match time-wise rows of Kd.
nrow(dat$kd); nrow(dat$es)

#Use spc.timeMatch() to get row indexes of Es that would match those of Kd time-wise
t_idx=spc.timeMatch((dat$kd), (dat$es))
#Verification
all(time(dat$es)[t_idx]==time(dat$kd))
```

---

spc.updateheader *Update a field of the @header slot of a* Spectra *object*

Description

Updates or changes the value of a field in the header slot of Spectra object

Usage

```
spc.updateheader(object,Name,value,...)

## S4 method for signature 'Spectra'
spc.updateheader(object, Name, value, ...)

## S4 method for signature 'list'
spc.updateheader(object, Name, value)
```

Arguments

object	A Spectra object
Name	of the header field to be updated
value	to update header with
...	arguments to be passed to or from other methods

Examples

```
sp=spc.example_spectra()
sp@header
sp <- spc.updateheader(sp,"Station", 11)
sp@header

#SpcList example
sp=spc.example_spectra()
BL=spc.makeSpcList(sp,"CAST")
BL[[1]]@header
BL[[1]] <- spc.updateheader(BL[[1]],"Station", 11)
BL[[1]]@header
```

SpcHeader-class SpcHeader *class for header object storing metadata.*

Description

Definition for SpcHeader. This class is required for the @header slot of Spectra object. This class directly inherits R lists, so there is no additional slots.

Examples

```
new("SpcHeader")
```

SpcHeaderAdd *Set a field of the @header slot of a* SpcHeader *class object*

Description

Function add the value of a field in the header slot of SpcHeader class object

Usage

```
SpcHeaderAdd (object,Name,Value,...)

## S4 method for signature 'SpcHeader'
SpcHeaderAdd(object, Name, Value)
```

Arguments

object	of class SpcHeader
Name	a character variable
Value	a numeric variable
...	arguments to be passed to or from other methods

Examples

```
sp=spc.example_spectra()
sp@header
sp@ShortName
sp@header=SpcHeaderAdd(sp@header,sp@ShortName,10)
sp@header
```

SpcHeaderList-class SpcHeaderList *class.*

Description

Definition for SpcHeaderList. This class provides a collection of multiple SpcHeader objects inside a list.

Examples

```
h1 = new("SpcHeader")
h2 = new("SpcHeader")
as(list(h1, h2), "SpcHeaderList")
new("SpcHeaderList")
```

SpcList *Constructor function for the* SpcList *class.*

Description

With this function, it is easy to create a SpcList object given an list containing multipls Spectra objects.

Usage

```
SpcList(x)
```

Arguments

x	a list object

Examples

```
sp=spc.example_spectra()
#Create an SpcList object using two Spectra objects
as(list(sp,sp^2), "SpcList")
#the above is the same as
SpcList(list(sp,sp^2))
```

SpcList-class *SpcList class definition*

Description

Definition for SpcList, a class to store multiple Spectra objects inside a list-like object. See the help of the constructor function SpcList.

Slots

.Data list, Inherited R list object

by character, Determines the header field in the Spectra objects within the SpcList that describes how they are different one from the other.

Spectra *Constructor function for the class* Spectra.

Description

Spectra Creates an instance of class Spectra.

Usage

```
Spectra(inDF, Spectra, Wavelengths, Units, space, time, endTime, header, ...)
```

Arguments

inDF	a long-format data.frame containing LAT,LON and TIME columns as well as Ancillary data. See stConstruct for more information on long DF format.
Spectra	matrix containing spectral data. Channels are in columns, observations are in rows. If Spectra is missing, the first length(Wavelengths) columns of inDF will be taken as spectral data.
Wavelengths	numeric vector containing wavelengths of spectral channels.
Units	character defining the units of the wavelengths.
space	a character or integer holding the column index in inDF where the spatial coordinates are (if length(space)==2) or where the ID of the spatial location is (if (length(space)==1). If space is not provided, inDF columns are searched to match one of the following : LAT,lat,latitude,LATITUDE,LON,LONG,lon,long,longitude,LONGITUDE If LAT & LON are not found, they set the dummy value of 1.
time	character or integer indicating the column in inDF containing POSIXct TIME data values. if time is missing, it is set the dummy integer sequential vector of 1:nrow(Spectra).
endTime	character or integer indicating the column in inDF containing POSIXct ENDTIME data values. If the temporal measurements are performed over an interval, time and endtime contain the time for the start and end of intervals respectively. If the temporal measurements are performed over a time-instance, then endTime==TIME. If endTime is not provided, inDF columns are searched to match ENDTIME. If none found, then it is assumed that data are time-instance measurements. For more information, see the documentation of **spacetime**.
header	SpcHeader object containing metadata
...	other input arguments to be passed to the new() function

Details

This constructor function uses The function Spectra() calls spacetime::stConstruct() that is the constructor of the STIDF class using an input data.frame object of long-table format.

length{@Wavelengths}==ncol(@Spectra). The default @WavelengthsUnit is nm^-1.

Value

Returns an object of class Spectra.

Examples

```
fnm = file.path(base::system.file(package = "geoSpectral"),
"test_data","particulate_absorption.csv.gz")
fnm=gsub("\\\\", "/", fnm)
abs = read.table(fnm,sep=",",header=TRUE)
abs$STATION=factor(abs$STATION)
abs[1:2,1:17] #Display only the first 2 rows and first 17 columns if the data frame
lbd = as.numeric(gsub("X","",colnames(abs)[14:514]))
Units="1/m"
colnames(abs)= gsub("X",paste("anap","_",sep=""), colnames(abs))
colnames(abs)= gsub("PRES","DEPTH", colnames(abs))
abs = abs[,c(14:514,1:13)] #Rearrange so that Spectra columns come first
tz<-strsplit(as.character(abs$TIME)," ")[[1]][[3]] #Extract the timezone
abs$TIME = as.POSIXct(as.character(abs$TIME),tz=tz) #Compute the time

#Space and time columns are automatically found in the column names of inDF
myS<-Spectra(abs,Wavelengths=lbd,Units=Units,ShortName="a_nap")

#Space and time columns are explicitly chosen from inDF columns
myS<-Spectra(abs,Wavelengths=lbd, space=c("LONG","LAT"), time="TIME",
Units=Units,ShortName="a_nap")
```

Spectra-class *Spectra class definition*

Description

Spectra class is the main class provided by the package geoSpectRal. It allows storage of spectral or non-spectra data with space and time attributes.

Slots

ShortName character, A short name for the parameter described in the spectra object.

LongName character, A long name for the parameter described in the spectra object.

Spectra matrix, n by m matrix, describing n rows of spectral data (or time) in m channels (columns).

data data.frame n by t data frame, describind n rows of ancillary data of t variables. This slot is inherited from STIDF class.

Wavelengths numeric vector, length of m. Wavelength data.

WavelengthsUnit character, Units of the @Wavelength slot

header SpcHeader, Header object. See SpcHeader-class.

Units character, Units of spectral data.

`UnitsAnc` character, Units of each column of the @data slot holding ancillary data.

`ShortNameAnc` character, A short name for each column of the @data slot holding ancillary data.

`LongNameAnc` character, A long name for each column of the @data slot holding ancillary data.

`InvalidIdx` logical, length of m. Row index for measurements marked by the user as invalid.

`SelectedIdx` logical, length of m. Row index for measurements marked by the user as selected.

`ClassVersion` numeric, Version of the class.

`Spectra-coerce` *Conversion between* `Spectra` *and data.frame objects*

Description

Converting `Spectra` object to data.frame is straightforward while the conversion in the opposite direction requires a set of attributes to be present in the source data.frame object. These attributes are generally created during the conversion of a `Spectra` object into data.frame, they can also be manually set if they are non-existant (see the example below).

Arguments

`from`	The input object
`to`	Name of the class of output object

Examples

```
#Convert a Spectra object to data.frame
sp <- spc.example_spectra()
df <- as(sp, "data.frame")
class(df); dim(df)
attributes(df)

#Convert the data.frame back to Spectra
sp2 <- as(df, "Spectra")

#Convert a bare data.frame to Spectra with minimal attributes
df2 <- data.frame(ch1=c(1,2,3,4), ch2=c(5,6,7,8), TIME=Sys.time()+1:4, LAT=1:4, LON=5:8)
attr(df2, "Units") <- "m-1"
attr(df2, "Wavelengths") <- c(500, 600)
attr(df2, "ShortName") <- "abs"
as(df2, "Spectra")
```

subset,SpcList-method *Subsetting for a* spcList *and Spectra classes*

Description

Subsetting can be achieved using the implementation of the R function subset() for Spectra and SpcList classes. It is possible to perform a row-wise selection. The argument "select" is not implemented yet. Use "[]".

Usage

```
## S4 method for signature 'SpcList'
subset(x, subset, select, drop = FALSE, ...)
```

Arguments

x	A Spectra object.
subset	logical expression indicating elements or rows to keep: missing values are taken as false.
select	expression, indicating columns to select from the Spectra object.
drop	passed on to [indexing operator. Default is FALSE .
...	arguments to be passed to or from other methods.

Examples

```
fnm = file.path(system.file(package = "geoSpectral"), "test_data","particulate_absorption.csv.gz")
abs = read.table(fnm,sep=",",header=TRUE)
abs$STATION=factor(abs$STATION)
abs[1:2,1:17] #Display only the first 2 rows and first 17 columns if the data frame
lbd = as.numeric(gsub("X","",colnames(abs)[14:514]))
Units="1/m"
colnames(abs)= gsub("X",paste("anap","_",sep=""), colnames(abs))
colnames(abs)= gsub("PRES","DEPTH", colnames(abs))
abs = abs[,c(14:514,1:13)]
tz<-strsplit(as.character(abs$TIME)," ")[[1]][[3]] #Extract the timezone
abs$TIME = as.POSIXct(as.character(abs$TIME),tz=tz)
myS<-Spectra(abs,Wavelengths=lbd,Units=Units,ShortName="a_nap")
myS
head(spc.getwavelengths(myS))
spc.setwavelengths(myS) <- 300:800
myS[1:10]
myS[,"anap_400"]
myS[,c("anap_400","anap_500")]
myS[1:10,30:50] #Selection of channels by column index
lbd = as.numeric(c(412,440,490,555,670))
myS[1:10,lbd] #Selection of channels by wavelength
myS[1:10,"415::450"]
myS$CAST #Returns Ancillary data
```

```
myS$anap_400 #Returns spectra as numeric vector
head(myS[["anap_400"]]) #Returns spectra as numeric vector
head(myS[[c("Snap","Offset")]]) #Returns data.frame
#Subsetting rows with respect to the value of Ancillary data
subset(myS,DEPTH<=30)
#Subsetting rows with respect to the value of Spectral data
subset(myS,anap_440<=0.01)
#Selecting Ancillary data columns, leaving Spectral columns intact
subset(myS,subset=DEPTH<=30,select="CAST")
```

---

subset,Spectra-method *Subsetting for a* Spectra *and spcList classes*

Description

Subsetting can be achieved using the implementation of the R function subset() for Spectra and SpcList classes It is possible to perform a row-wise selection

Usage

```
## S4 method for signature 'Spectra'
subset(x, subset, select, drop = FALSE, ...)
```

Arguments

x	A Spectra object
subset	logical expression indicating elements or rows to keep: missing values are taken as false.
select	Condition selected
drop	passed on to [indexing operator. Default is FALSE
...	arguments to be passed to or from other methods.

Examples

```
fnm = file.path(system.file(package = "geoSpectral"), "test_data","particulate_absorption.csv.gz")
abs = read.table(fnm,sep=",",header=TRUE)
abs$STATION=factor(abs$STATION)
abs[1:2,1:17] #Display only the first 2 rows and first 17 columns if the data frame
lbd = as.numeric(gsub("X","",colnames(abs)[14:514]))
Units="1/m"
colnames(abs)= gsub("X",paste("anap","_",sep=""), colnames(abs))
colnames(abs)= gsub("PRES","DEPTH", colnames(abs))
abs = abs[,c(14:514,1:13)]
tz<-strsplit(as.character(abs$TIME)," ")[[1]][[3]] #Extract the timezone
abs$TIME = as.POSIXct(as.character(abs$TIME),tz=tz)
myS<-Spectra(abs,Wavelengths=lbd,Units=Units,ShortName="a_nap")
myS
```

```
head(spc.getwavelengths(myS))
spc.setwavelengths(myS) <- 300:800
myS[1:10]
myS[,"anap_400"]
myS[,c("anap_400","anap_500")]
myS[1:10,30:50] #Selection of channels by column index
lbd = as.numeric(c(412,440,490,555,670))
myS[1:10,lbd] #Selection of channels by wavelength
myS[1:10,"415::450"]
myS$CAST #Returns Ancillary data
myS$anap_400 #Returns spectra as numeric vector
head(myS[["anap_400"]]) #Returns spectra as numeric vector
head(myS[[c("Snap","Offset")]]) #Returns data.frame
#Subsetting rows with respect to the value of Ancillary data
subset(myS,DEPTH<=30)
#Subsetting rows with respect to the value of Spectral data
subset(myS,anap_440<=0.01)
#Selecting Ancillary data columns, leaving Spectral columns intact
subset(myS,subset=DEPTH<=30,select="CAST")
```

`$,SpcList-method` *Extract or replace parts of a* `SpcList` *object*

Description

Operators acting on `Spectra` objects to extract or replace parts

Usage

```
## S4 method for signature 'SpcList'
x$name
```

Arguments

`x`	A `Spectra` object from which to extract element(s) or in which to replace element(s)
`name`	A character (column name)

Examples

```
sp<-spc.example_spectra()
BL = spc.makeSpcList(sp,"STATION")

#Extract station 394 (returns Spectra object)
BL$`394`

BL@by="CRUISE"
BL[[1]]$CRUISE="Cruise1"
```

```
BL[[2]]$CRUISE="Cruise2"
BL[[3]]$CRUISE="Cruise3"
BL[[4]]$CRUISE="Cruise4"
names(BL)
BL$Cruise4
```

`$,Spectra-method` *Extract or replace parts of a* `Spectra` *object*

Description

Operators acting on `Spectra` objects to extract parts

Replace parts of a `Spectra` object

Operators acting on `Spectra` object and `Spectra` lists to extract or replace parts.

Usage

```
## S4 method for signature 'Spectra'
x$name

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

## S4 method for signature 'Spectra'
x[i, j]

## S4 method for signature 'Spectra,character,missing'
x[[i, j]]

## S4 replacement method for signature 'Spectra,character,missing'
x[[i, j]] <- value
```

Arguments

`x`	A `Spectra` object from which to extract element(s) or in which to replace element(s)
`name`	A character (column name) or a numeric (column index) variable
`value`	A vector or matrix or data.frame. Values to be replaced with matched `Spectra` column.
`i`	A numeric (row index) variable
`j`	A character (column name) or a numeric (column index) variable

Details

These operators are generic. You can write methods to handle indexing of specific classes of objects

Examples

```
 sp<-spc.example_spectra()
 # spc.colnames() is used to extract column names
 head(spc.colnames(sp))
 head(sp$anap_300)
 sp[,"anap_345"]
 sp[,"anap_345"] #returns Spectra object with only one channel (column)
 sp[1:3,"anap_345"] #returns Spectra object with first 3 rows and only one channel (column)
 # spc.colnames() is used to extract column names
 head(spc.colnames(sp))
 head(sp$anap_300)
 sp[,"anap_345"]
sp=spc.example_spectra()
sp #501 spectral channels in columns and 26 observations in rows
sp[1] #returns Spectra object, 501 spectral channels in columns and 1 observations in rows
names(sp)
sp[["CAST"]] #returns the CAST data column
sp[[4]] #returns the CAST data column
sp[["CAST"]]=12 #Modify the CAST column
sp[["CAST"]] #returns the CAST data column
```

Index