Package ‘OpenSpecy’

November 26, 2023

Type Package

Title Analyze, Process, Identify, and Share Raman and (FT)IR Spectra

Version 1.0.6

Date 2023-11-25

Description Raman and (FT)IR spectral analysis tool for plastic particles and other environmental samples (Cowger et al. 2021, <doi:10.1021/acs.analchem.1c00123>). With read_any(), Open Specy provides a single function for reading individual, batch, or map spectral data files like .asp, .csv, .jdx, .spec, .spa, .0, and .zip. process_spec() simplifies processing spectra, including smoothing, baseline correction, range restriction and flattening, intensity conversions, wavenumber alignment, and min-max normalization. Spectra can be identified in batch using an onboard reference library (Cowger et al. 2020, <doi:10.1177/0003702820929064>) using match_spec(). A Shiny app is available via run_app() or online at <https://openanalysis.org/openspecy/>.


BugReports https://github.com/wincowgerDEV/OpenSpecy-package/issues/

License CC BY 4.0

Encoding UTF-8

LazyLoad true

LazyData true

VignetteBuilder knitr

Depends R (>= 4.1.0)

Imports methods, data.table, jsonlite, yaml, osfr, caTools, hyperSpec, imager, plotly, digest, signal, glmnet, shiny

Suggests knitr, rmarkdown, testthat (>= 3.1.9), config, qs, shinyjs, shinyWidgets, bs4Dash, dplyr, ggplot2, DT, curl, aws.s3, mongolite, loggit

RoxygenNote 7.2.3
Config/testthat/edition  3

NeedsCompilation no

Author  Win Cowger [cre, aut, dtc] (<https://orcid.org/0000-0001-9226-3104>),
Zacharias Steinmetz [aut] (<https://orcid.org/0000-0001-6675-5033>),
Nick Leong [aut] (<https://orcid.org/0009-0008-3313-4132>),
Andrea Faltynkova [aut, dtc] (<https://orcid.org/0000-0003-2523-3137>),
Andrew B Gray [ctb] (<https://orcid.org/0000-0003-2252-3137>),
Hannah Hapich [ctb] (<https://orcid.org/0000-0003-0000-6632>),
Jennifer Lynch [ctb, dtc] (<https://orcid.org/0000-0003-3572-8782>),
Hannah De Frond [ctb, dtc] (<https://orcid.org/0000-0003-1199-0727>),
Keenan Munno [ctb, dtc] (<https://orcid.org/0000-0003-2916-5944>),
Chelsa Rochman [ctb, dtc] (<https://orcid.org/0000-0002-7624-711X>),
Sebastian Primpeke [ctb, dtc] (<https://orcid.org/0000-0001-7633-8524>),
Orestis Herodotou [ctb],
Mary C Norris [ctb],
Christine M Knauss [ctb] (<https://orcid.org/0000-0003-4404-8922>),
Aleksandra Karapetrova [ctb, dtc, rev] (<https://orcid.org/0000-0002-9856-1644>),
Vesna Teofilovic [ctb] (<https://orcid.org/0000-0003-5571-4820>),
Laura A. T. Markley [ctb] (<https://orcid.org/0000-0003-0620-8366>),
Shreyas Patankar [ctb, dtc],
Rachel Kozloski [ctb, dtc] (<https://orcid.org/0000-0001-1211-9351>),
Samiksha Singh [ctb],
Katherine Lasdin [ctb],
Cristiane Vidal [ctb] (<https://orcid.org/0000-0001-6363-9475>),
Clare Murphy-Hagan [ctb] (<https://orcid.org/0009-0009-9629-2856>),
Philipp Baumann [ctb] (<https://orcid.org/0000-0002-3194-8975>),
Pierre Roudier [ctb],
National Renewable Energy Laboratory [fnd],
Possibility Lab [fnd]

Maintainer  Win Cowger <wincowger@gmail.com>

Repository  CRAN

Date/Publication  2023-11-25 23:50:02 UTC

R topics documented:

  adj_intens ..................................................  3
  adj_res ....................................................  4
  as_OpenSpecy ...............................................  6
  check_lib ..................................................  9
  collapse_spec .............................................. 12
  conform_spec ............................................... 14
  cor_spec .................................................... 15
  c_spec ...................................................... 18
  head.OpenSpecy ............................................. 19
  human_ts .................................................... 20
**adj_intens**

Adjust spectral intensities to absorbance units.

**Description**

Converts reflectance or transmittance intensity units to absorbance units.

**Usage**

adj_intens(x, ...)

## Default S3 method:
adj_intens(x, ...)

## S3 method for class 'OpenSpecy'
adj_intens(x, type = "none", make_rel = TRUE, ...)

**Arguments**

- **x**
  - a list object of class OpenSpecy.

- **type**
  - a character string specifying whether the input spectrum is in absorbance units ("none", default) or needs additional conversion from "reflectance" or "transmittance" data.

- **make_rel**
  - logical; if TRUE spectra are automatically normalized with make_rel().

- **...**
  - further arguments passed to submethods; this is to adj_neg() for adj_intens() and to conform_res() for conform_intens().
Details

Many of the Open Specy functions will assume that the spectrum is in absorbance units. For example, see `subtr_baseline()`. To run those functions properly, you will need to first convert any spectra from transmittance or reflectance to absorbance using this function. The transmittance adjustment uses the $\log(1/T)$ calculation which does not correct for system and particle characteristics. The reflectance adjustment uses the Kubelka-Munk equation $(1 - R)^2/2R$. We assume that the reflectance intensity is a percent from 1-100 and first correct the intensity by dividing by 100 so that it fits the form expected by the equation.

Value

`adj_intens()` returns a data frame containing two columns named "wavenumber" and "intensity".

Author(s)

Win Cowger, Zacharias Steinmetz

See Also

`subtr_baseline()` for spectral background correction.

Examples

data("raman_hdpe")

adj_intens(raman_hdpe)

adj_res

Normalization and conversion of spectral data

Description

`adj_res()` and `conform_res()` are helper functions to align wavenumbers in terms of their spectral resolution. `adj_neg()` converts numeric intensities $y < 1$ into values $\geq 1$, keeping absolute differences between intensity values by shifting each value by the minimum intensity. `make_rel()` converts intensities $y$ into relative values between 0 and 1 using the standard normalization equation. If `na.rm` is TRUE, missing values are removed before the computation proceeds.

Usage

`adj_res(x, res = 1, fun = round)`

`conform_res(x, res = 5)`

`adj_neg(y, na.rm = FALSE)`

`mean_replace(y, na.rm = TRUE)`
adj_res

is_empty_vector(x)

Arguments

x a numeric vector or an \texttt{R} object which is coercible to one by \texttt{as.vector}(x, "numeric"); \texttt{x} should contain the spectral wavenumbers.

res spectral resolution supplied to \texttt{fun}.

fun the function to be applied to each element of \texttt{x}; defaults to \texttt{round()} to round to a specific resolution \texttt{res}.

y a numeric vector containing the spectral intensities.

na.rm logical. Should missing values be removed?

Details

\texttt{adj_res()} and \texttt{conform_res()} are used in Open Specy to facilitate comparisons of spectra with different resolutions. \texttt{adj_neg()} is used to avoid errors that could arise from log transforming spectra when using \texttt{adj_intens()} and other functions. \texttt{make_rel()} is used to retain the relative height proportions between spectra while avoiding the large numbers that can result from some spectral instruments.

Value

\texttt{adj_res()} and \texttt{conform_res()} return a numeric vector with resolution-conformed wavenumbers. \texttt{adj_neg()} and \texttt{make_rel()} return numeric vectors with the normalized intensity data.

Author(s)

Win Cowger, Zacharias Steinmetz

See Also

\texttt{min()} and \texttt{round()}; \texttt{adj_intens()} for log transformation functions; \texttt{conform_spec()} for conforming wavenumbers of an \texttt{OpenSpecy} object to be matched with a reference library

Examples

\[
\begin{align*}
\text{adj_res} & \quad \text{seq}(500, 4000, 4), 5) \\
\text{conform_res} & \quad \text{seq}(500, 4000, 4)) \\
\text{adj_neg} & \quad \text{c(-1000, -1, 0, 1, 10))} \\
\text{make_rel} & \quad \text{c(-1000, -1, 0, 1, 10))}
\end{align*}
\]
as_OpenSpecy

Create OpenSpecy objects

Description

Functions to check if an object is an OpenSpecy, or coerce it if possible.

Usage

as_OpenSpecy(x, ...)

## S3 method for class 'openSpecy'
as_OpenSpecy(x, session_id = FALSE, ...)

## S3 method for class 'list'
as_OpenSpecy(x, ...)

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

## S3 method for class 'data.frame'
as_OpenSpecy(x, colnames = list(wavenumber = NULL, spectra = NULL), ...)

## Default S3 method:
as_OpenSpecy(
  x,
  spectra,
  metadata = list(file_name = NULL, user_name = NULL, contact_info = NULL, organization = NULL, citation = NULL, spectrum_type = NULL, spectrum_identity = NULL, material_form = NULL, material_phase = NULL, material_producer = NULL, material_purity = NULL, material_quality = NULL, material_color = NULL, material_other = NULL, cas_number = NULL, instrument_used = NULL, instrument_accessories = NULL, instrument_mode = NULL, intensity_units = NULL, spectral_resolution = NULL, laser_light_used = NULL, number_of_accumulations = NULL, total_acquisition_time_s = NULL, data_processing_procedure = NULL, level_of_confidence_in_identification = NULL, other_info = NULL, license = "CC BY-NC"),
  attributes = list(intensity_unit = NULL, derivative_order = NULL, baseline = NULL, spectra_type = NULL),
  coords = "gen_grid",
  session_id = FALSE,
  ...
)

is_OpenSpecy(x)

check_OpenSpecy(x)
OpenSpecy(x, ...)  

gen_grid(n)

**Arguments**

x 
depending on the method, a list with all OpenSpecy parameters, a vector with the wavenumbers for all spectra, or a data.frame with a full spectrum in the classic Open Specy format.

session_id 
logical. Whether to add a session ID to the metadata. The session ID is based on current session info so metadata of the same spectra will not return equal if session info changes. Sometimes that is desirable.

colnames 
names of the wavenumber column and spectra column, makes assumptions based on column names or placement if NULL.

spectra 
spectral intensities formatted as a data.table with one column per spectrum.

metadata 
metadata for each spectrum with one row per spectrum, see details.

attributes 
a list of attributes describing critical aspects for interpreting the spectra. see details.

coords 
spatial coordinates for the spectra.

n 
number of spectra to generate the spatial coordinate grid with.

... 
additional arguments passed to submethods.

**Details**

as_OpenSpecy() converts spectral datasets to a three part list; the first with a vector of the wavenumbers of the spectra, the second with a data.table of all spectral intensities ordered as columns, the third item is another data.table with any metadata the user provides or is harvested from the files themselves.

The metadata argument may contain a named list with the following details (* = minimum recommended).

file_name* The file name, defaults to basename() if not specified

user_name* User name, e.g. "Win Cowger"

contact_info Contact information, e.g. "1-513-673-8956, wincowger@gmail.com"

organization Affiliation, e.g. "University of California, Riverside"

citation Data citation, e.g. "Primpke, S., Wirth, M., Lorenz, C., & Gerdts, G. (2018). Reference database design for the automated analysis of microplastic samples based on Fourier transform infrared (FTIR) spectroscopy. Analytical and Bioanalytical Chemistry. doi:10.1007/s00216-0181156x"

spectrum_type* Raman or FTIR

spectrum_identity* Material/polymer analyzed, e.g. "Polystyrene"

material_form Form of the material analyzed, e.g. textile fiber, rubber band, sphere, granule

material_phase Phase of the material analyzed (liquid, gas, solid)
as_OpenSpecy

material_producer  Producer of the material analyzed, e.g. Dow
material_purity  Purity of the material analyzed, e.g. 99.98%
material_quality  Quality of the material analyzed, e.g. consumer product, manufacturer material, analytical standard, environmental sample
material_color  Color of the material analyzed, e.g. blue, #0000ff, (0, 0, 255)
material_other  Other material description, e.g. 5 µm diameter fibers, 1 mm spherical particles
cas_number  CAS number, e.g. 9003-53-6
instrument_used  Instrument used, e.g. Horiba LabRam
instrument_accessories  Instrument accessories, e.g. Focal Plane Array, CCD
instrument_mode  Instrument modes/settings, e.g. transmission, reflectance
intensity_units*  Units of the intensity values for the spectrum, options transmittance, reflectance, absorbance
spectral_resolution  Spectral resolution, e.g. 4/cm
laser_light_used  Wavelength of the laser/light used, e.g. 785 nm
number_of_accumulations  Number of accumulations, e.g 5
total_acquisition_time_s  Total acquisition time (s), e.g. 10 s
data_processing_procedure  Data processing procedure, e.g. spikefilter, baseline correction, none
level_of_confidence_in_identification  Level of confidence in identification, e.g. 99%
other_info  Other information
license  The license of the shared spectrum; defaults to "CC BY-NC" (see https://creativecommons.org/licenses/by-nc/4.0/ for details). Any other creative commons license is allowed, for example, CC0 or CC BY
session_id  A unique user and session identifier; populated automatically with paste(digest(Sys.info()), digest(sessionInfo()), sep = "/")
file_id  A unique file identifier; populated automatically with digest(object[c("wavenumber", "spectra")])

The attributes argument may contain a named list with the following details, when set, they will be used to automate transformations and warning messages:

intensity_units  supported options include "absorbance", "transmittance", or "reflectance"
derivative_order  supported options include "0", "1", or "2"
baseline  supported options include "raw" or "nobaseline"
spectra_type  supported options include "ftir" or "raman"

Value

as_OpenSpecy() and OpenSpecy() returns three part lists described in details. is_OpenSpecy() returns TRUE if the object is an OpenSpecy and FALSE if not. gen_grid() returns a data.table with x and y coordinates to use for generating a spatial grid for the spectra if one is not specified in the data.
Author(s)

Zacharias Steinmetz, Win Cowger

See Also

read_spec() for reading OpenSpecy objects.

Examples

data("raman_hdpe")

# Inspect the spectra
raman_hdpe # see how OpenSpecy objects print.
raman_hdpe$wavenumber # look at just the wavenumbers of the spectra.
raman_hdpe$spectra # look at just the spectral intensities data.table.
raman_hdpe$metadata # look at just the metadata of the spectra.

# Creating a list and transforming to OpenSpecy
as_OpenSpecy(list(wavenumber = raman_hdpe$wavenumber,
                  spectra = raman_hdpe$spectra,
                  metadata = raman_hdpe$metadata[,,-c("x", "y")]))

# If you try to produce an OpenSpecy using an OpenSpecy it will just return
# the same object.
as_OpenSpecy(raman_hdpe)

# Creating an OpenSpecy from a data.frame
as_OpenSpecy(x = data.frame(wavenumber = raman_hdpe$wavenumber,
                             spectra = raman_hdpe$spectra$spectra$intensity))

# Test that the spectrum is formatted as an OpenSpecy object.
is_OpenSpecy(raman_hdpe)
is_OpenSpecy(raman_hdpe$spectra)

Description

These functions will import the spectral libraries from Open Specy if they were not already downloaded. The CRAN does not allow for deployment of large datasets so this was a workaround that we are using to make sure everyone can easily get Open Specy functionality running on their desktop. Please see the references when using these libraries. These libraries are the accumulation of a massive amount of effort from independant groups and each should be attributed when you are using their data.
Usage

check_lib(
  type = c("derivative", "nobaseline", "raw", "mediod", "model"),
  path = "system",
  condition = "warning"
)

get_lib(
  type = c("derivative", "nobaseline", "raw", "mediod", "model"),
  path = "system",
  node = "x7dpz",
  conflicts = "overwrite",
  ...
)

load_lib(type, path = "system")

rm_lib(
  type = c("derivative", "nobaseline", "raw", "mediod", "model"),
  path = "system"
)

Arguments

type      library type to check/retrieve; defaults to c("derivative", "nobaseline",
                      "raw", "mediod", "model") which reads everything.
path      where to save or look for local library files; defaults to "system" pointing to
          system.file("extdata", package = "OpenSpecy").
condition determines if check_lib() should warn ("warning", the default) or throw an
          error ("error").
node      the OSF node to be retrieved; should be "x7dpz" unless you maintain your own
          OSF node with spectral libraries.
conflicts determines what happens when a file with the same name exists at the specified
          destination. Can be one of the following (see osf_download() for details):
          "error" throw an error and abort the file transfer operation.
          "skip" skip the conflicting file(s) and continue transferring the remaining files.
          "overwrite" (default) replace the existing file with the transferred copy.
          ...
          further arguments passed to osf_download().

Details

check_lib() checks to see if the Open Specy reference library already exists on the users
computer. get_lib() downloads the Open Specy library from OSF (doi:10.17605/OSF.IO/X7DPZ).
load_lib() will load the library into the global environment for use with the Open Specy func-
tions. rm_lib() removes the libraries from your computer.
check_lib

Value

check_lib() and get_lib() return messages only; load_lib() returns an OpenSpecy object containing the respective spectral reference library.

Author(s)

Zacharias Steinmetz, Win Cowger

References


**Further contribution of spectra:** Suja Sukumaran (Thermo Fisher Scientific), Aline Carvalho, Jennifer Lynch (NIST), Claudia Cella and Dora Mehn (JRC), Horiba Scientific, USDA Soil Characterization Data ([https://ncsslabdatamart.sc.egov.usda.gov](https://ncsslabdatamart.sc.egov.usda.gov)), Archaeometrielabor, and S.B. Engelsen (Royal Vet. and Agricultural University, Denmark). Kimmel Center data was collected and provided by Prof. Steven Weiner (Kimmel Center for Archaeological Science, Weizmann Institute of Science, Israel).

### Examples

```r
## Not run:
check_lib("derivative")
get_lib("derivative")
spec_lib <- load_lib("derivative")

## End(Not run)
```

### collapse_spec

**Define features**

Functions for analyzing features, like particles, fragments, or fibers, in spectral map oriented OpenSpecy object.
Usage

collapse_spec(x, ...)

## Default S3 method:
collapse_spec(x, ...)

## S3 method for class 'OpenSpecy'
collapse_spec(x, ...)

def_features(x, ...)

## Default S3 method:
def_features(x, ...)

## S3 method for class 'OpenSpecy'
def_features(x, features, ...)

Arguments

x an OpenSpecy object
features a logical vector or character vector describing which of the spectra are of features (TRUE) and which are not (FALSE). If a character vector is provided, it should represent the different feature types present in the spectra.
...
additional arguments passed to subfunctions.

Details
def_features() accepts an OpenSpecy object and a logical or character vector describing which pixels correspond to particles. collapse_spec() takes an OpenSpecy object with particle-specific metadata (from def_features()) and collapses the spectra to median intensities for each unique particle. It also updates the metadata with centroid coordinates, while preserving the feature information on area and Feret max.

Value

An OpenSpecy object appended with metadata about the features or collapsed for the features.

Author(s)

Win Cowger, Zacharias Steinmetz

Examples

tiny_map <- read_extdata("CA_tiny_map.zip") |> read_any()
identified_map <- def_features(tiny_map, tiny_map$metadata$x == 0)
collapse_spec(identified_map)
**conform_spec**

Conform spectra to a standard wavenumber series

---

**Description**

Spectra can be conformed to a standard suite of wavenumbers to be compared with a reference library or to be merged to other spectra.

**Usage**

```r
conform_spec(x, ...)  
## Default S3 method:  
conform_spec(x, ...)  
## S3 method for class 'OpenSpecy'  
conform_spec(x, range = NULL, res = 5, type = "interp", ...)  
```

**Arguments**

- `x` a list object of class `OpenSpecy`
- `range` a vector of new wavenumber values, can be just supplied as a min and max value.
- `res` spectral resolution adjusted to or NULL if the raw range should be used.
- `type` the type of wavenumber adjustment to make. "interp" results in linear interpolation while "roll" conducts a nearest rolling join of the wavenumbers.
- `...` further arguments passed to `approx()`

**Value**

`adj_intens()` returns a data frame containing two columns named "wavenumber" and "intensity"

**Author(s)**

Win Cowger, Zacharias Steinmetz

**See Also**

`restrict_range()` and `flatten_range()` for adjusting wavenumber ranges; `subtr_baseline()` for spectral background correction

**Examples**

```r
data("raman_hdpe")  
conform_spec(raman_hdpe, c(1000, 2000))
```
Identify and filter spectra

Description

match_spec() joins two OpenSpecy objects and their metadata based on similarity. cor_spec() correlates two OpenSpecy objects, typically one with knowns and one with unknowns. ident_spec() retrieves the top match values from a correlation matrix and formats them with metadata. get_metadata() retrieves metadata from OpenSpecy objects. max_cor_named() formats the top correlation values from a correlation matrix as a named vector. filter_spec() filters an Open Specy object.

Usage

cor_spec(x, ...)

## Default S3 method:
cor_spec(x, ...)

## S3 method for class 'OpenSpecy'
cor_spec(x, library, na.rm = T, conform = F, type = "roll", ...)

match_spec(x, ...)

## Default S3 method:
match_spec(x, ...)

## S3 method for class 'OpenSpecy'
match_spec(
  x,
  library,
  na.rm = T,
  conform = F,
  type = "roll",
  top_n = NULL,
  order = NULL,
  add_library_metadata = NULL,
  add_object_metadata = NULL,
  fill = NULL,
  ...
)

ident_spec(
  cor_matrix,
  x,
  library,
  top_n = NULL,
  add_library_metadata = NULL,
  ...
add_object_metadata = NULL, ...
)

get_metadata(x, ...)

## Default S3 method:
get_metadata(x, ...)

## S3 method for class 'OpenSpecy'
get_metadata(x, logic, rm_empty = TRUE, ...)

max_cor_named(cor_matrix, na.rm = T)

filter_spec(x, ...)

## Default S3 method:
filter_spec(x, ...)

## S3 method for class 'OpenSpecy'
filter_spec(x, logic, ...)

ai_classify(x, ...)

## Default S3 method:
ai_classify(x, ...)

## S3 method for class 'OpenSpecy'
ai_classify(x, library, fill = NULL, ...)

Arguments

x an OpenSpecy object, typically with unknowns.
library an OpenSpecy or glmnet object representing the reference library of spectra or model to use in identification.
na.rm logical; indicating whether missing values should be removed when calculating correlations. Default is TRUE.
conform Whether to conform the spectra to the library wavenumbers or not.
type the type of conformation to make returned by conform_spec()
top_n integer; specifying the number of top matches to return. If NULL (default), all matches will be returned.
order an OpenSpecy used for sorting, ideally the unprocessed one; NULL skips sorting.
add_library_metadata name of a column in the library metadata to be joined; NULL if you don’t want to join.
add_object_metadata name of a column in the object metadata to be joined; NULL if you don’t want to join.
fill an OpenSpecy object with a single spectrum to be used to fill missing values for alignment with the AI classification.

cor_matrix a correlation matrix for object and library, can be returned by cor_spec()

logic a logical or numeric vector describing which spectra to keep.

rm_empty logical; whether to remove empty columns in the metadata.

... additional arguments passed cor().

Value

match_spec() and ident_spec() will return a data.table-class() containing correlations between spectra and the library. The table has three columns: object_id, library_id, and match_val. Each row represents a unique pairwise correlation between a spectrum in the object and a spectrum in the library. If top_n is specified, only the top top_n matches for each object spectrum will be returned. If add_library_metadata is is.character, the library metadata will be added to the output. If add_object_metadata is is.character, the object metadata will be added to the output. filter_spec() returns an OpenSpecy object. cor_spec() returns a correlation matrix. get_metadata() returns a data.table-class() with the metadata for columns which have information.

Author(s)

Win Cowger, Zacharias Steinmetz

See Also

adj_intens() converts spectra; get_lib() retrieves the Open Specy reference library; load_lib() loads the Open Specy reference library into an R object of choice

Examples

data("test_lib")

unknown <- read_extdata("ftir_ldpe_soil.asp") |> read_any() |> conform_spec(range = test_lib$wavenumber, res = spec_res(test_lib)) |> process_spec() cor_spec(unknown, test_lib)

match_spec(unknown, test_lib, add_library_metadata = "sample_name", top_n = 1)
c_spec

Manage spectral objects

Description

c_spec() concatenates OpenSpecy objects. sample_spec() samples spectra from an OpenSpecy object.

Usage

c_spec(x, ...)

## Default S3 method:
c_spec(x, ...)

## S3 method for class 'OpenSpecy'
c_spec(x, ...)

## S3 method for class 'list'
c_spec(x, range = NULL, res = 5, ...)

sample_spec(x, ...)

## Default S3 method:
sample_spec(x, ...)

## S3 method for class 'OpenSpecy'
sample_spec(x, size = 1, prob = NULL, ...)

Arguments

x a list of OpenSpecy objects.
range a numeric providing your own wavenumber ranges or character argument called "common" to let c_spec() find the common wavenumber range of the supplied spectra. NULL will interpret the spectra having all the same wavenumber range.
res defaults to NULL, the resolution you want the output wavenumbers to be.
size the number of spectra to sample.
prob probabilities to use for the sampling.
... further arguments passed to submethods.

Value

c_spec() and sample_spec() return OpenSpecy objects.

Author(s)

Zacharias Steinmetz, Win Cowger
See Also

conform_spec() for conforming wavenumbers

Examples

# Concatenating spectra
spectra <- lapply(c(read_extdata("raman_hdpe.csv"),
                   read_extdata("ftir_ldpe_soil.asp")), read_any)
common <- c_spec(spectra, range = "common", res = 5)
range <- c_spec(spectra, range = c(1000, 2000), res = 5)

# Sampling spectra
tiny_map <- read_any(read_extdata("CA_tiny_map.zip"))
sampled <- sample_spec(tiny_map, size = 3)
## Arguments

- `x`: an OpenSpecy object.
- `...`: further arguments passed to the respective default method.

## Details

`head()` shows the first few lines of an OpenSpecy object. `print()` prints the contents of an OpenSpecy object. `summary()` produces a result summary of an OpenSpecy object. `plot()` produces a `matplot()` of an OpenSpecy object; `lines()` adds new spectra to it.

## Value

`head()`, `print()`, and `summary()` return a textual representation of an OpenSpecy object. `plot()` and `lines()` return a plot. `as.data.frame()` and `as.data.table()` convert OpenSpecy objects into tabular data.

## Author(s)

Zacharias Steinmetz, Win Cowger

## See Also

`head()`, `print()`, `summary()`, `matplot()`, and `matlines()`, `as.data.frame()`, `as.data.table()`

## Examples

```r
data("raman_hdpe")

# Printing the OpenSpecy object
print(raman_hdpe)

# Displaying the first few lines of the OpenSpecy object
head(raman_hdpe)

# Plotting the spectra
plot(raman_hdpe)
```

---

## Description

This helper function creates human readable timestamps in the form of `%%Y%m%d-%H%M%S` at the current time.
Usage

human_ts()

Details

Human readable timestamps are appended to file names and fields when metadata are shared with the Open Specy community.

Value

human_ts() returns a character value with the respective timestamp.

Author(s)

Win Cowger, Zacharias Steinmetz

See Also

format.Date for date conversion functions

Examples

human_ts()

make_rel

---

Description

make_rel() converts intensities x into relative values between 0 and 1 using the standard normalization equation. If na.rm is TRUE, missing values are removed before the computation proceeds.

Usage

make_rel(x, ...)

## Default S3 method:
make_rel(x, na.rm = FALSE, ...)

## S3 method for class 'OpenSpecy'
make_rel(x, na.rm = FALSE, ...)

Arguments

x a numeric vector or an R OpenSpecy object
na.rm logical. Should missing values be removed?
... further arguments passed to make_rel().
Details

make_rel() is used to retain the relative height proportions between spectra while avoiding the large numbers that can result from some spectral instruments.

Value

make_rel() return numeric vectors (if vector provided) or an OpenSpecy object with the normalized intensity data.

Author(s)

Win Cowger, Zacharias Steinmetz

See Also

min() and round(); adj_intens() for log transformation functions; conform_spec() for conforming wavenumbers of an OpenSpecy object to be matched with a reference library

Examples

make_rel(c(-1000, -1, 0, 1, 10))

plotly_spec

Interactive plots for OpenSpecy objects

Description

These functions generate heatmaps, spectral plots, and interactive plots for OpenSpecy data.

Usage

plotly_spec(x, ...)

## Default S3 method:
plotly_spec(x, ...)

## S3 method for class 'OpenSpecy'
plotly_spec(
x,
x2 = NULL,
line = list(color = "rgb(255, 255, 255)"),
line2 = list(dash = "dot", color = "rgb(255,0,0)"),
font = list(color = "#FFFFFF"),
plot_bgc = "rgba(17, 0, 73, 0)",
paper_bgc = "rgb(0, 0, 0)",
showlegend = FALSE,
Arguments

x       an OpenSpecy object containing metadata and spectral data for the first group.
x2      an optional second OpenSpecy object containing metadata and spectral data for
the second group.

- **line**
  - list; line parameter for x; passed to `add_trace()`.

- **line2**
  - list; line parameter for x2; passed to `add_trace()`.

- **font**
  - list; passed to `layout()`.

- **plotbgcolor**
  - color value; passed to `layout()`.

- **paperbgcolor**
  - color value; passed to `layout()`.

- **showlegend**
  - whether to show the legend passed to `plot_ly()`.

- **z**
  - optional numeric vector specifying the intensity values for the heatmap. If not provided, the function will use the intensity values from the OpenSpecy object.

- **sn**
  - optional numeric value specifying the signal-to-noise ratio threshold. If provided along with `min_sn`, regions with SNR below the threshold will be excluded from the heatmap.

- **cor**
  - optional numeric value specifying the correlation threshold. If provided along with `min_cor`, regions with correlation below the threshold will be excluded from the heatmap.

- **min_sn**
  - optional numeric value specifying the minimum signal-to-noise ratio for inclusion in the heatmap. Regions with SNR below this threshold will be excluded.

- **min_cor**
  - optional numeric value specifying the minimum correlation for inclusion in the heatmap. Regions with correlation below this threshold will be excluded.

- **select**
  - optional index of the selected spectrum to highlight on the heatmap.

- **colorscale**

- **...**
  - further arguments passed to `plot_ly()`.

**Value**

A plotly heatmap object displaying the OpenSpecy data. A subplot containing the heatmap and spectra plot. A plotly object displaying the spectra from the OpenSpecy object(s).

**Author(s)**

Win Cowger, Zacharias Steinmetz

**Examples**

data("raman_hdpe")
tiny_map <- read_extdata("CA_tiny_map.zip") |> read_zip()plotly_spec(raman_hdpe)

heatmap_spec(tiny_map, z = tiny_map$metadata$y, showlegend = TRUE)
sample_spec(tiny_map, size = 12) |>interactive_plot(select = 2, x2 = raman_hdpe)
process_spec

Description

process_spec() is a monolithic wrapper function for all spectral processing steps.

Usage

process_spec(x, ...)

## Default S3 method:
process_spec(x, ...)

## S3 method for class 'OpenSpecy'
process_spec(
x,
  active = TRUE,
  adj_intens = FALSE,
  adj_intens_args = list(type = "none"),
  conform_spec = TRUE,
  conform_spec_args = list(range = NULL, res = 5, type = "interp"),
  restrict_range = FALSE,
  restrict_range_args = list(min = 0, max = 6000),
  flatten_range = FALSE,
  flatten_range_args = list(min = 2200, max = 2420),
  subtr_baseline = FALSE,
  subtr_baseline_args = list(type = "polynomial", degree = 8, raw = FALSE, baseline = NULL),
  smooth_intens = TRUE,
  smooth_intens_args = list(polynomial = 3, window = 11, derivative = 1, abs = TRUE),
  make_rel = TRUE,
  make_rel_args = list(na.rm = TRUE),
  ...
)

Arguments

x an OpenSpecy object.
active logical; indicating whether to perform processing. If TRUE, the processing steps will be applied. If FALSE, the original data will be returned.
adj_intens logical; describing whether to adjust the intensity units.
adj_intens_args named list of arguments passed to smooth_intens().
conform_spec logical; whether to conform the spectra to a new wavenumber range and resolution.
process_spec

conform_spec_args  named list of arguments passed to conform_spec().
restrict_range    logical; indicating whether to restrict the wavenumber range of the spectra.
restrict_range_args named list of arguments passed to restrict_range().
flatten_range     logical; indicating whether to flatten the range around the carbon dioxide region.
flatten_range_args named list of arguments passed to flatten_range().
subtr_baseline    logical; indicating whether to subtract the baseline from the spectra.
subtr_baseline_args named list of arguments passed to subtr_baseline().
smooth_intens     logical; indicating whether to apply a smoothing filter to the spectra.
smooth_intens_args named list of arguments passed to smooth_intens().
make_rel          logical; if TRUE spectra are automatically normalized with make_rel().
make_rel_args     named list of arguments passed to make_rel().
na.rm             Whether to allow NA or set all NA values to
                  further arguments passed to subfunctions.

Value

process_spec() returns an OpenSpecy object with processed spectra based on the specified parameters.

Examples

data("raman_hdpe")
plot(raman_hdpe)

# Process spectra with range restriction and baseline subtraction
process_spec(raman_hdpe,
             restrict_range = TRUE,
             restrict_range_args = list(min = 500, max = 3000),
             subtr_baseline = TRUE,
             subtr_baseline_args = list(type = "polynomial",
                                        polynomial = 8)) |> 
lines(col = "darkred")

# Process spectra with smoothing and derivative
process_spec(raman_hdpe,
             smooth_intens = TRUE,
             smooth_intens_args = list(
                 polynomial = 3,
                 window = 11,
                 derivative = 1
             ) ) |> 
lines(col = "darkgreen")
Description

Raman spectrum of high-density polyethylene (HDPE) provided by Horiba Scientific.

Format

An threepart list of class OpenSpecy containing:

- **wavenumber**: spectral wavenumbers [1/cm] (vector of 964 rows)
- **spectra**: absorbance values - (a data.table with 964 rows and 1 column)
- **metadata**: spectral metadata

Author(s)

Zacharias Steinmetz, Win Cowger

References


Examples

```r
data(raman_hdpe)
print(raman_hdpe)
```

---

**read_any**

*Read spectral data from multiple files*

Description

Wrapper functions for reading files in batch.

Usage

```r
read_any(file, ...)
read_zip(file, ...)
```
Arguments

file file to be read from or written to.
... further arguments passed to the submethods.

Details

read_any() provides a single function to quickly read in any of the supported formats, it assumes that the file extension will tell it how to process the spectra. read_zip() provides functionality for reading in spectral map files with ENVI file format or as individual files in a zip folder. If individual files, spectra are concatenated.

Value

All read_*() functions return OpenSpecy objects

Author(s)

Zacharias Steinmetz, Win Cowger

See Also

read_spec()

Examples

read_extdata("raman_hdpe.csv") |> read_any()
read_extdata("ftir_ldpe_soil.asp") |> read_any()
read_extdata("testdata_zipped.zip") |> read_zip()
read_extdata("CA_tiny_map.zip") |> read_zip()

---

read_envi

Read ENVI data

Description

This function allows ENVI data import.

Usage

read_envi(
  file,
  header = NULL,
  share = NULL,
  metadata = list(file_name = basename(file), user_name = NULL, contact_info = NULL,
  organization = NULL, citation = NULL, spectrum_type = NULL, spectrum_identity = NULL,
  material_form = NULL, material_phase = NULL, material_producer = NULL,
)
material_purity = NULL, material_quality = NULL, material_color = NULL,
material_other = NULL, cas_number = NULL, instrument_used = NULL,
instrument_accessories = NULL, instrument_mode = NULL, spectral_resolution = NULL,
laser_light_used = NULL, number_of_accumulations = NULL,

total_acquisition_time_s = NULL, data_processing_procedure = NULL,
level_of_confidence_in_identification = NULL, other_info = NULL, license = "CC BY-NC"),
...
}

Arguments

file  name of the binary file.
header 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 with .hdr ex-
       tension.
share  defaults to NULL; needed to share spectra with the Open Specy community; see
       share_spec() for details.
metadata  a named list of the metadata; see as_OpenSpecy() for details.
...  further arguments passed to the submethods.

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 via read.ENVI().

Value

An OpenSpecy object.

Author(s)

Zacharias Steinmetz, Claudia Beleites

See Also

read_spec() for reading .y(a)ml, .json, or .rds (OpenSpecy) files; read_text(), read_asp(),
read_spa(), read_spc(), and read_jdx() for text files, .asp, .spa, .spa, .spc, and .jdx formats,
respectively; read_opus() for reading .0 (OPUS) files; read_zip() and read_any() for wrapper
functions; read.ENVI()
read_opus

Read spectral data from Bruker OPUS binary files

Description

Read file(s) acquired with a Bruker Vertex FTIR Instrument. This function is basically a fork of
opus_read() from https://github.com/pierreroudier/opusreader.

Usage

read_opus(
  file,
  share = NULL,
  metadata = list(file_name = basename(file), user_name = NULL, contact_info = NULL,
                 organization = NULL, citation = NULL, spectrum_type = NULL, spectrum_identity = NULL,
                 material_form = NULL, material_phase = NULL, material_producer = NULL,
                 material_purity = NULL, material_quality = NULL, material_color = NULL,
                 material_other = NULL, cas_number = NULL, instrument_used = NULL,
                 instrument_accessories = NULL, instrument_mode = NULL, spectral_resolution = NULL,
                 laser_light_used = NULL, number_of_accumulations = NULL,
                 total_acquisition_time_s = NULL, data_processing_procedure = NULL,
                 level_of_confidence_in_identification = NULL, other_info = NULL, license =
                 "CC BY-NC"),
  type = "spec",
  digits = 1L,
  atm_comp_minus4offset = FALSE)

Arguments

file character vector with path to file(s).
share defaults to NULL; needed to share spectra with the Open Specy community; see
share_spec() for details.
metadata a named list of the metadata; see as_OpenSpecy() for details.
type character vector of spectra types to extract from OPUS binary file. Default
is "spec", which will extract the final spectra, e.g. expressed in absorbance
(named AB in Bruker OPUS programs). Possible additional values for the char-
acter vector supplied to type are "spec_no_atm_comp" (spectrum of the sample
without compensation for atmospheric gases, water vapor and/or carbon
dioxide), "sc_sample" (single channel spectrum of the sample measurement),
"sc_ref" (single channel spectrum of the reference measurement), "ig_sample"
(interferogram of the sample measurement) and "ig_ref" (interferogram of the
reference measurement).
digits Integer that specifies the number of decimal places used to round the wavenum-
bers (values of x-variables).
atm_comp_minus4offset

Logical whether spectra after atmospheric compensation are read with an offset of -4 bytes from Bruker OPUS files; default is FALSE.

Details

The type of spectra returned by the function when using type = "spec" depends on the setting of the Bruker instrument: typically, it can be either absorbance or reflectance.

The type of spectra to extract from the file can also use Bruker’s OPUS software naming conventions, as follows:

- ScSm corresponds to sc_sample
- ScRf corresponds to sc_ref
- IgSm corresponds to ig_sample
- IgRf corresponds to ig_ref

Value

An OpenSpecy object.

Author(s)

Philipp Baumann, Zacharias Steinmetz, Win Cowger

See Also

read_spec() for reading .y(a)ml, .json, or .rds (OpenSpecy) files; read_text(), read_asp(), read_spa(), read_spc(), and read_jdx() for text files, .asp, .spa, .spa, .spc, and .jdx formats, respectively; read_text() for reading .dat (ENVI) files; read_zip() and read_any() for wrapper functions; read_opus_raw();

Examples

read_extdata("ftir_ps.0") |> read_opus()
Arguments

- **rw**: a raw vector
- **type**: character vector of spectra types to extract from OPUS binary file. Default is "spec", which will extract the final spectra, e.g. expressed in absorbance (named AB in Bruker OPUS programs). Possible additional values for the character vector supplied to type are "spec_no_atm_comp" (spectrum of the sample without compensation for atmospheric gases, water vapor and/or carbon dioxide), "sc_sample" (single channel spectrum of the sample measurement), "sc_ref" (single channel spectrum of the reference measurement), "ig_sample" (interferogram of the sample measurement) and "ig_ref" (interferogram of the reference measurement).
- **atm_comp_minus4offset**: logical; whether spectra after atmospheric compensation are read with an offset of -4 bytes from Bruker OPUS files. Default is FALSE.

Details

The type of spectra returned by the function when using **type = "spec"** depends on the setting of the Bruker instrument: typically, it can be either absorbance or reflectance.

The type of spectra to extract from the file can also use Bruker’s OPUS software naming conventions, as follows:

- ScSm corresponds to sc_sample
- ScRf corresponds to sc_ref
- IgSm corresponds to ig_sample
- IgRf corresponds to ig_ref

Value

A list of 10 elements:

- **metadata**: a data.frame containing metadata from the OPUS file.
- **spec**: if "spec" was requested in the type option, a matrix of the spectrum of the sample (otherwise set to NULL).
- **spec_no_atm_comp**: if "spec_no_atm_comp" was requested in the type option, a matrix of the spectrum of the sample without atmospheric compensation (otherwise set to NULL).
- **sc_sample**: if "sc_sample" was requested in the type option, a matrix of the single channel spectrum of the sample (otherwise set to NULL).
- **sc_ref**: if "sc_ref" was requested in the type option, a matrix of the single channel spectrum of the reference (otherwise set to NULL).
- **ig_sample**: if "ig_sample" was requested in the type option, a matrix of the interferogram of the sample (otherwise set to NULL).
- **ig_ref**: if "ig_ref" was requested in the type option, a matrix of the interferogram of the reference (otherwise set to NULL).
wavenumbers if "spec" or "spec_no_atm_comp" was requested in the type option, a numeric vector of the wavenumbers of the spectrum of the sample (otherwise set to NULL).

wavenumbers_sc_sample if "sc_sample" was requested in the type option, a numeric vector of the wavenumbers of the single channel spectrum of the sample (otherwise set to NULL).

wavenumbers_sc_ref if "sc_ref" was requested in the type option, a numeric vector of the wavenumbers of the single channel spectrum of the reference (otherwise set to NULL).

**Author(s)**

Philipp Baumann and Pierre Roudier

**See Also**

read_opus()

---

**Description**

Functions for reading spectral data from external file types. Currently supported reading formats are .csv and other text files, .asp, .spa, .spc, and .jdx. Additionally, .0 (OPUS) and .dat (ENVI) files are supported via read_opus() and read_envi(), respectively. read_zip() takes any of the files listed above. Note that proprietary file formats like .0, .asp, and .spa are poorly supported but will likely still work in most cases.

**Usage**

```r
read_text(
  file,
  colnames = NULL,
  method = "fread",
  share = NULL,
  metadata = list(file_name = basename(file), user_name = NULL, contact_info = NULL,
                  organization = NULL, citation = NULL, spectrum_type = NULL, spectrum_identity = NULL,
                  material_form = NULL, material_phase = NULL, material_producer = NULL,
                  material_purity = NULL, material_quality = NULL, material_color = NULL,
                  material_other = NULL, cas_number = NULL, instrument_used = NULL,
                  instrument_accessories = NULL, instrument_mode = NULL, spectral_resolution = NULL,
                  laser_light_used = NULL, number_of_accumulations = NULL,
                  total_acquisition_time_s = NULL, data_processing_procedure = NULL,
                  level_of_confidence_in_identification = NULL, other_info = NULL, license = 
                  "CC BY-NC"),
  ...
)
```
read.asp(
  file,
  share = NULL,
  metadata = list(file_name = basename(file), user_name = NULL, contact_info = NULL,
                 organization = NULL, citation = NULL, spectrum_type = NULL, spectrum_identity = NULL,
                 material_form = NULL, material_phase = NULL, material_producer = NULL,
                 material_purity = NULL, material_quality = NULL, material_color = NULL,
                 material_other = NULL, cas_number = NULL, instrument_used = NULL,
                 instrument_accessories = NULL, instrument_mode = NULL, spectral_resolution = NULL,
                 laser_light_used = NULL, number_of_accumulations = NULL,
                 total_acquisition_time_s = NULL, data_processing_procedure = NULL,
                 level_of_confidence_in_identification = NULL, other_info = NULL, license = "CC BY-NC"),
...
)

read.spa(
  file,
  share = NULL,
  metadata = list(file_name = basename(file), user_name = NULL, contact_info = NULL,
                 organization = NULL, citation = NULL, spectrum_type = NULL, spectrum_identity = NULL,
                 material_form = NULL, material_phase = NULL, material_producer = NULL,
                 material_purity = NULL, material_quality = NULL, material_color = NULL,
                 material_other = NULL, cas_number = NULL, instrument_used = NULL,
                 instrument_accessories = NULL, instrument_mode = NULL, spectral_resolution = NULL,
                 laser_light_used = NULL, number_of_accumulations = NULL,
                 total_acquisition_time_s = NULL, data_processing_procedure = NULL,
                 level_of_confidence_in_identification = NULL, other_info = NULL, license = "CC BY-NC"),
...
)

read.spc(
  file,
  share = NULL,
  metadata = list(file_name = basename(file), user_name = NULL, contact_info = NULL,
                 organization = NULL, citation = NULL, spectrum_type = NULL, spectrum_identity = NULL,
                 material_form = NULL, material_phase = NULL, material_producer = NULL,
                 material_purity = NULL, material_quality = NULL, material_color = NULL,
                 material_other = NULL, cas_number = NULL, instrument_used = NULL,
                 instrument_accessories = NULL, instrument_mode = NULL, spectral_resolution = NULL,
                 laser_light_used = NULL, number_of_accumulations = NULL,
                 total_acquisition_time_s = NULL, data_processing_procedure = NULL,
                 level_of_confidence_in_identification = NULL, other_info = NULL, license = "CC BY-NC"),
read_text

...}

read_jdx(
  file,
  share = NULL,
  metadata = list(file_name = basename(file), user_name = NULL, contact_info = NULL,
                   organization = NULL, citation = NULL, spectrum_type = NULL, spectrum_identity = NULL,
                   material_form = NULL, material_phase = NULL, material_producer = NULL,
                   material_purity = NULL, material_quality = NULL, material_color = NULL,
                   material_other = NULL, cas_number = NULL, instrument_used = NULL,
                   instrument_accessories = NULL, instrument_mode = NULL, spectral_resolution = NULL,
                   laser_light_used = NULL, number_of_accumulations = NULL,
                   total_acquisition_time_s = NULL, data_processing_procedure = NULL,
                   level_of_confidence_in_identification = NULL, other_info = NULL, license =
                   "CC BY-NC"),
...
)

read_extdata(file = NULL)

Arguments

file file to be read from or written to.
colnames character vector of length = 2 indicating the column names for the wavenumber
and intensity; if NULL columns are guessed.
method submethod to be used for reading text files; defaults to fread() but read.csv()
works as well.
share defaults to NULL; needed to share spectra with the Open Specy community; see
share_spec() for details.
metadata a named list of the metadata; see as_OpenSpecy() for details.
... further arguments passed to the submethods.

Details

read_spc() and read_jdx() are wrappers around the functions provided by the hyperSpec. Other
functions have been adapted various online sources. Metadata is harvested if possible. There are
many unique iterations of spectral file formats so there may be bugs in the file conversion. Please
contact us if you identify any.

Value

All read_*() functions return data frames containing two columns named “wavenumber” and
“intensity”.

Author(s)

Zacharias Steinmetz, Win Cowger
restrict_range

See Also

read_spec() for reading .y(a)ml, .json, or .rds (OpenSpecy) files; read_opus() for reading .0 (OPUS) files; read_envi() for reading .dat (ENVI) files; read_zip() and read_any() for wrapper
terms; read.jdx(); read.spc()  

Examples

read_extdata("raman_hdpe.csv") |> read_text()
read_extdata("raman_atacamit.spc") |> read_spc()
read_extdata("ftir_ldpe_soil.asp") |> read_asp()
read_extdata("testdata_zipped.zip") |> read_zip()

restrict_range

Range restriction and flattening for spectra

Description

restrict_range() restricts wavenumber ranges to user specified values. Multiple ranges can be
specified by inputting a series of max and min values in order. flatten_range() will flatten ranges
of the spectra that should have no peaks. Multiple ranges can be specified by inputting the series of
max and min values in order.

Usage

restrict_range(x, ...)

## Default S3 method:
restrict_range(x, ...)

## S3 method for class 'OpenSpecy'
restrict_range(x, min, max, make_rel = TRUE, ...)

flatten_range(x, ...)

## Default S3 method:
flatten_range(x, ...)

## S3 method for class 'OpenSpecy'
flatten_range(x, min = 2200, max = 2400, make_rel = TRUE, ...)

Arguments

x an OpenSpecy object.
min a vector of minimum values for the range to be flattened.
max a vector of maximum values for the range to be flattened.
make_rel logical; should the output intensities be normalized to the range \([0, 1]\) using `make_rel()` function?

... additional arguments passed to subfunctions; currently not in use.

Value

An OpenSpecy object with the spectral intensities within specified ranges restricted or flattened.

Author(s)

Win Cowger, Zacharias Steinmetz

See Also

`conform_spec()` for conforming wavenumbers to be matched with a reference library; `adj_intens()` for log transformation functions; `min()` and `round()`

Examples

```r

```test_noise <- as_OpenSpecy(x = seq(400,4000, by = 10),
  spectra = data.frame(intensity = rnorm(361)))

plot(test_noise)

restrict_range(test_noise, min = 1000, max = 2000)

flattened_intensities <- flatten_range(test_noise, min = c(1000, 2000),
  max = c(1500, 2500))

plot(flattened_intensities)
```

run_app

Run Open Specy app

Description

This wrapper function starts the graphical user interface of Open Specy.

Usage

```r

run_app(path = "system", log = TRUE, ref = "main", test_mode = FALSE, ...)
```

Arguments

- **path**
  to store the downloaded app files; defaults to "system" pointing to `system.file(package = "OpenSpecy")`.

- **log**
  logical; enables/disables logging to `tempdir()`

- **ref**
  git reference; could be a commit, tag, or branch name. Defaults to "main". Only change this in case of errors.

- **test_mode**
  logical; for internal testing only.

- ... arguments passed to `runApp()`.
details

After running this function the Open Specy GUI should open in a separate window or in your computer browser.

value

This function normally does not return any value, see `runGitHub()`.

Author(s)

Zacharias Steinmetz

See Also

`runGitHub()`

examples

```r
## Not run:
run_app()

## End(Not run)
```

share_spec

**Share data with the Open Specy community**

Description

This helper function shares spectral data and metadata with the Open Specy community.

Please note: that `share_spec()` only provides basic sharing functionality if used interactively. This means that files are only formatted and saved for sharing but are not sent automatically. This only works with hosted instances of Open Specy.

Usage

```r
share_spec(x, ...)
```

## Default S3 method:

```r
share_spec(x, ...)
```

## S3 method for class 'OpenSpecy'

```r
share_spec(x, file = NULL, share = "system", credentials = NULL, ...)
```
Arguments

- `x`: a list object of class OpenSpecy.
- `file`: file to share (optional).
- `share`: accepts any local directory to save the spectrum for later sharing via email to `<wincowger@gmail.com>`; "system" (default) uses the Open Specy package directory at `system.file("extdata",package = "OpenSpecy")`; if a correct API token exists, "cloud" shares the spectrum with the cloud.
- `credentials`: a named list of credentials for cloud sharing; required if `share = "cloud"`.
- `...`: further arguments passed to the submethods.

Value

`share_spec()` returns only messages/warnings.

Author(s)

Zacharias Steinmetz, Win Cowger

See Also

`read_text()`; `digest()`; `sessionInfo()`

Examples

```r
## Not run:
data("raman_hdpe")
share_spec(raman_hdpe,
    metadata = list(
        user_name = "Win Cowger",
        spectrum_type = "FTIR",
        spectrum_identity = "PE",
        license = "CC BY-NC"
    ))

## End(Not run)
```

Description

This function calculates common signal and noise metrics for OpenSpecy objects.
Usage

    sig_noise(x, ...)

## Default S3 method:
    sig_noise(x, ...)

## S3 method for class 'OpenSpecy'
    sig_noise(
        x,
        metric = "run_sig_over_noise",
        na.rm = TRUE,
        step = 20,
        sig_min = NULL,
        sig_max = NULL,
        noise_min = NULL,
        noise_max = NULL,
        abs = T,
        ...
    )

Arguments

x                      an OpenSpecy object.
metric               character; specifying the desired metric to calculate.
na.rm                logical; indicating whether missing values should be removed when calculating signal and noise. Default is TRUE.
step                numeric; the step size of the region to look for the run_sig_over_noise option.
sig_min            numeric; the minimum wavenumber value for the signal region.
sig_max            numeric; the maximum wavenumber value for the signal region.
noise_min          numeric; the minimum wavenumber value for the noise region.
noise_max          numeric; the maximum wavenumber value for the noise region.
abs                logical; whether to return the absolute value of the result Options include "sig" (mean intensity), "noise" (standard deviation of intensity), "sig_times_noise" (absolute value of signal times noise), "sig_over_noise" (absolute value of signal / noise), "run_sig_over_noise" (absolute value of signal / noise where signal is estimated as the max intensity and noise is estimated as the height of a low intensity region.), "log_tot_sig" (sum of the inverse log intensities, useful for spectra in log units), or "tot_sig" (sum of intensities).
...

Value

A numeric vector containing the calculated metric for each spectrum in the OpenSpecy object.

See Also

    restrict_range()
smooth_intens

Examples

data("raman_hdpe")

sig_noise(raman_hdpe, metric = "sig")
sig_noise(raman_hdpe, metric = "noise")
sig_noise(raman_hdpe, metric = "sig_times_noise")

smooth_intens Smooth spectral intensities

Description

This smoother can enhance the signal to noise ratio of the data and uses a Savitzky-Golay filter with a running window of data points and the polynomial specified.

Usage

smooth_intens(x, ...)

## Default S3 method:
smooth_intens(x, ...)

## S3 method for class 'OpenSpecy'
smooth_intens(
  x,
  polynomial = 3,
  window = 11,
  derivative = 1,
  abs = TRUE,
  make_rel = TRUE,
  ...
)

Arguments

x an object of class OpenSpecy.

polynomial polynomial order for the filter

window number of data points in the window, filter length (must be odd).

derivative the derivative order if you want to calculate the derivative. Zero (default) is no derivative.

abs logical; whether you want to calculate the absolute value of the resulting output.

make_rel logical; if TRUE spectra are automatically normalized with make_rel().

... further arguments passed to sgolay().
Details

This is a wrapper around the filter function in the signal package to improve integration with other Open Specy functions. A typical good smooth can be achieved with 11 data point window and a 3rd or 4th order polynomial.

Value

smooth_intens() returns an OpenSpecy object.

Author(s)

Win Cowger, Zacharias Steinmetz

References


See Also

sgolay()

Examples

data("raman_hdpe")

smooth_intens(raman_hdpe)
subtr_baseline

Arguments

  x  a numeric vector with wavenumber data or an OpenSpecy object.
  ... further arguments passed to subfunctions; currently not used.

Details

  The spectral resolution is the minimum wavenumber, wavelength, or
  frequency difference between two lines in a spectrum that can still be distinguished.

Value

  spec_res() returns a single numeric value.

Author(s)

  Win Cowger, Zacharias Steinmetz

Examples

  data("raman_hdpe")
  spec_res(raman_hdpe)

Description

  This baseline correction routine iteratively finds the baseline of a spectrum using a polynomial
  fitting or accepts a manual baseline.

Usage

  subtr_baseline(x, ...)

    # Default S3 method:
    subtr_baseline(x, ...)

    # S3 method for class 'OpenSpecy'
    subtr_baseline(
      x,
      type = "polynomial",
      degree = 8,
      raw = FALSE,
      baseline,
      make_rel = TRUE,
      ...
    )
Arguments

x a list object of class OpenSpecy.
type one of "polynomial" or "manual" depending on whether you want spectra to be corrected with a manual baseline or with polynomial baseline fitting.
degree the degree of the polynomial. Must be less than the number of unique points when raw is FALSE. Typically a good fit can be found with a 8th order polynomial.
raw if TRUE, use raw and not orthogonal polynomials.
baseline an OpenSpecy object containing the baseline data to be subtracted.
make_rel logical; if TRUE spectra are automatically normalized with make_rel().
... further arguments passed to poly().

Details

This is a translation of Michael Stephen Chen’s MATLAB code written for the imodpolyfit routine from Zhao et al. 2007.

Value

subtr_baseline() returns a data frame containing two columns named "wavenumber" and "intensity".

Author(s)

Win Cowger, Zacharias Steinmetz

References


See Also

poly(); smooth_intens()

Examples

data("raman_hdpe")

subtr_baseline(raman_hdpe)
Description

Reference library with 29 FTIR and 28 Raman spectra used for examples and internal testing.

Format

An OpenSpecy object; sample_name is the class of the spectra.

Author(s)

Win Cowger

Examples

data("test_lib")

write_spec | Read and write spectral data

Description

Functions for reading and writing spectral data to and from OpenSpecy format. OpenSpecy objects are lists with components wavenumber, spectra, and metadata. Currently supported formats are .y(a)ml, .json, or .rds.

Usage

write_spec(x, ...)

## Default S3 method:
write_spec(x, ...)

## S3 method for class 'OpenSpecy'
write_spec(x, file, method = NULL, digits = getOption("digits"), ...)

read_spec(file, share = NULL, method = NULL, ...)

as_hyperSpec(x)
Arguments

- `x` an object of class `OpenSpecy`.
- `file` file path to be read from or written to.
- `method` optional; function to be used as a custom reader or writer. Defaults to the appropriate function based on the file extension.
- `digits` number of significant digits to use when formatting numeric values; defaults to `getOption("digits")`.
- `share` defaults to `NULL`; needed to share spectra with the Open Specy community; see `share_spec()` for details.
- `...` further arguments passed to the submethods.

Details

Due to floating point number errors there may be some differences in the precision of the numbers returned if using multiple devices for .json and .yaml files but the numbers should be nearly identical. `readRDS()` should return the exact same object every time.

Value

`read_spec()` reads data formatted as an OpenSpecy object and returns a list object of class `OpenSpecy` containing spectral data. `write_spec()` writes a file for an object of class `OpenSpecy` containing spectral data. `as_hyperspec()` converts an OpenSpecy object to a `hyperSpec-class` object.

Author(s)

Zacharias Steinmetz, Win Cowger

See Also

`OpenSpecy()`, `read_text()`, `read_asp()`, `read_spa()`, `read_spc()`, and `read_jdx()` for text files, .asp, .spa, .spc, and .jdx formats, respectively; `read_zip()` and `read_any()` for wrapper functions; `saveRDS()`, `readRDS()`, `write_yaml()`, `read_yaml()`, `write_json()`, `read_json()`;

Examples

```r
read_extdata("raman_hdpe.yml") |> read_spec()
read_extdata("raman_hdpe.json") |> read_spec()
read_extdata("raman_hdpe.rds") |> read_spec()

## Not run:
data(raman_hdpe)
write_spec(raman_hdpe, "raman_hdpe.yml")
write_spec(raman_hdpe, "raman_hdpe.json")
write_spec(raman_hdpe, "raman_hdpe.rds")

# Convert an OpenSpecy object to a hyperSpec object
hyper <- as_hyperSpec(raman_hdpe)

## End(Not run)
```
Index

* data
  raman_hdpe, 27
  test_lib, 45
-
  add_trace, 24
  adj_intens, 3, 5, 17, 22, 37
  adj_neg, 3
  adj_neg(adj_res), 4
  adj_res, 4
  ai_classify(cor_spec), 15
  approx, 14
  as.data.frame, 20
  as.data.frame.OpenSpecy
    (head.OpenSpecy), 19
  as.data.table, 20
  as.data.table.OpenSpecy
    (head.OpenSpecy), 19
  as_hyperSpec(write_spec), 45
  as_OpenSpecy, 6, 29, 30, 35

  basename, 7
  c_spec, 18
  check_lib, 9
  check_OpenSpecy(as_OpenSpecy), 6
  collapse_spec, 12
  conform_res, 3
  conform_res(adj_res), 4
  conform_spec, 5, 14, 19, 22, 26, 37
  cor, 17
  cor_spec, 15
  data.table, 27
  def_features(collapse_spec), 12
  digest, 39
  filter_spec(cor_spec), 15
  flatten_range, 14, 26
  flatten_range(restrict_range), 36
  format.Date, 21

  fread, 35
  gen_grid(as_OpenSpecy), 6
  get_lib, 17
  get_lib(check_lib), 9
  get_metadata(cor_spec), 15
  getOption, 46
  head, 20
  head.OpenSpecy, 19
  heatmap_spec(plotly_spec), 22
  human_ts, 20
  hyperSpec, 35
  ident_spec(cor_spec), 15
  interactive_plot(plotly_spec), 22
  is_empty_vector(adj_res), 4
  is_OpenSpecy(as_OpenSpecy), 6
  layout, 24
  lines.OpenSpecy(head.OpenSpecy), 19
  load_lib, 17
  load_lib(check_lib), 9
  make_rel, 3, 21, 26, 41, 44
  match_spec(cor_spec), 15
  matlines, 20
  matplot, 20
  max_cor_named(cor_spec), 15
  mean_replace(adj_res), 4
  min, 5, 22, 37
  OpenSpecy, 27, 46
  OpenSpecy(as_OpenSpecy), 6
  osf_download, 10
  plot.OpenSpecy(head.OpenSpecy), 19
  plot_ly, 24
  plotly_spec, 22
  poly, 44
  print, 20

48
INDEX

print.OpenSpecy(head.OpenSpecy), 19  
process_spec, 25

raman_hdpe, 27
read.csv, 35
read.ENVI, 29
read.jdx, 36
read.spc, 36
read_any, 27, 29, 31, 36, 46
read.asp, 29, 31, 46
read.asp(read_text), 33
read_envi, 28, 33, 36
read_extdata(read_text), 33
read_jdx, 29, 31, 46
read_jdx(read_text), 33
read_json, 46
read_opus, 29, 30, 33, 36
read_opus_raw, 31, 31
read_spa, 29, 31, 46
read_spa(read_text), 33
read_spc, 29, 31, 46
read_spc(read_text), 33
read_spec, 9, 28, 29, 31, 36
read_spec(write_spec), 45
read_text, 29, 31, 33, 39, 46
read_yaml, 46
read_zip, 29, 31, 33, 36, 46
read_zip(read_any), 27
readRDS, 46
restrict_range, 14, 26, 36
restrict_range(), 40
rm_lib(check_lib), 9
round, 5, 22, 37
run_app, 37
runApp, 37
runGitHub, 38

sample_spec(c_spec), 18
saveRDS, 46
sessionInfo, 39
gsdlay, 41, 42
share_spec, 29, 30, 35, 38, 46
sig_noise, 39
smooth_intens, 25, 26, 41, 44
spec_res, 42
subtr_baseline, 4, 14, 26, 43
summary, 20
summary.OpenSpecy(head.OpenSpecy), 19

tempdir, 37
test_lib, 45
write_json, 46
write_spec, 45
write_yaml, 46