Package 'metabodecon'

January 21, 2025

Title Deconvolution and Alignment of 1d NMR Spectra

Version 1.2.6

Description A framework for deconvolution, alignment and postprocessing of 1-dimensional (1d) nuclear magnetic resonance (NMR) spectra, resulting in a data matrix of aligned signal integrals. The deconvolution part uses the algorithm described in Koh et al. (2009) <doi:10.1016/j.jmr.2009.09.003>. The alignment part is based on functions from the 'speaq' package, described in Beirnaert et al. (2018) <doi:10.1371/journal.pcbi.1006018> and Vu et al. (2011) <doi:10.1186/1471-2105-12-405>. A detailed description and evaluation of an early version of the package, 'MetaboDecon1D v0.2.2', can be found in Haeckl et al. (2021) <doi:10.3390/metabo11070452>.

License GPL (>= 3)

URL https://github.com/spang-lab/metabodecon/, https://spang-lab.github.io/metabodecon/

BugReports https://github.com/spang-lab/metabodecon/issues

biocViews NMR, Deconvolution

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Imports data.table, mathjaxr, readJDX, speaq, toscutil (>= 2.8.0), withr

Suggests devtools, digest, knitr, lifecycle, microbenchmark, pkgload, rmarkdown, usethis, vdiffr, testthat (>= 3.0.0)

LazyData true

Config/testthat/edition 3

Config/testthat/parallel true

Config/testthat/start-first read_spectrum, download_example_datasets, cache_example_datasets, deconvolute_ispecs, mcmapply, datadir, get_decon_params, generate_lorentz_curves, smooth_signals, speaq_align

RdMacros mathjaxr

BuildManual TRUE

Language en-US

NeedsCompilation no

Author Tobias Schmidt [aut, cre, cph],
Martina Haeckl [aut, cph],
Yanren Linda Hu [ctb],
Wolfram Gronwald [aut, cph]

Maintainer Tobias Schmidt <tobias.schmidt331@gmail.com>

Repository CRAN

Date/Publication 2025-01-21 16:00:06 UTC

Contents

`align` *Align Spectra*

Description

Align signals across a list of deconvoluted spectra using the 'CluPA' algorithm from the 'speaq' package, described in Beirnaert et al. (2018) doi:10.1371/journal.pcbi.1006018 and Vu et al. (2011) doi:10.1186/1471-2105-12-405 plus the additional peak combination described in `combine_peaks()`.

Usage

```
align(x, maxShift = 50, maxCombine = 5, verbose = FALSE)
```

Arguments

`x`	An object of type `decons1` or `decons2` as described in metabodecon_classes. To align `decons0` objects (as returned by the now deprecated MetaboDecon1D), you can use `as_decons2()` to convert it to a `decons2` object first.
`maxShift`	Maximum number of points along the "ppm-axis" a value can be moved by the 'speaq' package. 50 is a suitable starting value for plasma spectra with a digital resolution of 128K. Note that this parameter has to be individually optimized depending on the type of analyzed spectra and the digital resolution. For urine which is more prone to chemical shift variations this value most probably has to be increased. Passed as argument `maxShift` to `speaq_align()`.
`maxCombine`	Amount of adjacent columns which may be combined for improving the alignment. Passed as argument `range` to `combine_peaks()`.
`verbose`	Whether to print additional information during the alignment process.

Value

An object of type `align` as described in metabodecon_classes.

Examples

```
sim_dir <- metabodecon_file("bruker/sim")
spectra <- read_spectra(sim_dir)
decons <- deconvolute(spectra, sfr = c(3.55, 3.35))
aligned <- align(decons)
aligned
```

as_metabodecon_class *Convert to a Metabodecon Class*

Description

Convert an object to a Metabodecon class.

Usage

```
as_spectrum(x, sf = c(1000, 1e+06))

as_ispec(x, sf = c(1000, 1e+06))

as_idecon(x)

as_decon0(x, sf = NULL, spectrum = NULL, optional = TRUE)

as_decon1(
  x,
  sf = c(1000, 1e+06),
  spectrum = NULL,
  sfr = NULL,
  wshw = NULL,
  bwc = 2
)

as_decon2(
  x,
  sf = c(1000, 1e+06),
  spectrum = NULL,
  sfr = NULL,
  wshw = NULL,
  bwc = 2
)

as_spectra(
  x,
  file_format = "bruker",
  expno = 10,
  procno = 10,
  raw = FALSE,
  silent = TRUE,
  force = FALSE
)

as_ispecs(x, sf = c(1000, 1e+06))
```

```
as_idecons(x)

as_decons0(x, sfs = list(c(1000, 1e+06)), spectra = list(NULL), nworkers = 1)

as_decons1(
  x,
  sfs = list(c(1000, 1e+06)),
  spectra = list(NULL),
  sfrs = list(NULL),
  wshws = list(NULL),
  bwc = 2,
  nworkers = 1
)

as_decons2(
  x,
  sfs = list(c(1000, 1e+06)),
  spectra = list(NULL),
  sfrs = list(NULL),
  wshws = list(NULL),
  bwc = 2,
  nworkers = 1
)
```

Arguments

x	The object to convert.
sf	Scale factor. Only required if x is a decon0 object.
spectrum, spectra	The spectrum/spectra object corresponding to x as returned by read_spectrum() / read_spectra. Only required if x is a decon0 object.
optional	Logical. If TRUE, the two optional elements signal_free_region and range_water_signal_ppm are included in the returned decon0 object.
sfr, sfrs	sfr should be a vector specifying the borders of the signal free region. sfrs should be a list of such vectors. Only required if x is a decon0 object where element signal_free_region is missing (or a decons0 objected containing such decon0 objects).
wshw, wshws	wshw should specify the half width of the water signal region. wshws should be a list of such values. Only required if x is a decon0 object where element range_water_signal_ppm is missing (or a decons0 objected containing such decon0 objects).
bwc	Level of backwards compatibility. If bwc == 0, bug fixes introduced after version 0.2.2 of Metabodecon are not used. If bwc == 1, new features introduced after version 0.2.2 of Metabodecon (e.g. faster algorithms) are not used. If bwc == 2, all bug fixes and features introduced after version 0.2.2 are used. Support for bwc == 0 will be removed in 'metabodecon v2.0'.
file_format	The file_format of the spectrum file. E.g. "bruker" or "jcampdx".

expno, procno	The experiment/processing number for the file. E.g. "10". Only relevant if file_format equals "bruker". For details see section File Structure in the metabodecon FAQ.
raw	If FALSE, scales the returned signal intensities based on information available in the spectrum metadata, in particular NC_proc. For details see processing-reference.pdf, available at https://www.bruker.com/en.html at section 'Services & Support > Documentation & Manuals > Magnetic Resonance > Acquisition & Processing > TopSpin Processing Commands and Parameters' (requires login).
silent	If TRUE, no output will be printed to the console.
force	If TRUE, try to continue when encountering errors and print info messages instead. To hide these messages as well, set silent = TRUE.
sfs	List of scale factors. Only required if x is a list of decon0 objects.
nworkers	Number of workers for parallel processing.

Value

An object of the specified class.

Examples

```
dirpath <- metabodecon_file("sim_subset")
spectra <- read_spectra(dirpath)
spectrum <- spectra[[1]]
decons1 <- generate_lorentz_curves_sim(spectra)
decon1 <- generate_lorentz_curves_sim(spectrum)
decon2 <- as_decon2(decon1)
```

---

calculate_lorentz_curves

Calculate lorentz curves for each analyzed spectrum

Description

Calculates the lorentz curves of each investigated spectrum.

Martina Haeckl, 2021: Initial version. Tobias Schmidt, 2024: Minor updates to pass CRAN checks

Usage

```
calculate_lorentz_curves(deconv_result, number_of_files = NA)
```

Arguments

deconv_result	A list as returned by generate_lorentz_curves() or MetaboDecon1D.
number_of_files	Number of spectra to analyze

Value

If deconv_result holds the result of a single deconvolution, a matrix containing the generated Lorentz curves is returned, where each row depicts one Lorentz curve. If deconv_result is a list of deconvoluted spectra, a list of such matrices is returned.

See Also

MetaboDecon1D(), plot_triplets(), plot_lorentz_curves_save_as_png(), plot_spectrum_superposition_save_a

Examples

```
## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
## Deconvolute the spectra in folder "bruker/sim_subset" into a list of
## Lorentz Curves (specified via the parameters A, lambda and x_0).
## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
sim <- metabodecon_file("bruker/sim_subset")
decons <- generate_lorentz_curves_sim(sim)
decon0 <- decons[[1]]

## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
## Calculate the corresponding y values at each ppm value for each Lorentz
## Curve. I.e. you get a matrix of dimension n x m for each deconvolution,
## where n is the number of Lorentz Curves and m is the number of ppm values.
## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
yy <- calculate_lorentz_curves(decons)
y1 <- yy[[1]]
dim(y1)

## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
## Visualize the 5th, 9th and 11th Lorentz curve.
## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
nrs <- c(5, 9, 11)
col <- c("red", "blue", "orange")
desc <- paste("Lorentz curve", nrs)
plot(decon0$x_values_ppm, decon0$y_values, type = "l", lty = 2)
for (i in 1:3) lines(decon0$x_values_ppm, y1[nrs[i], ], col = col[i])
legend("topright", legend = desc, col = col, lty = 1)
```

---

combine_peaks *Combine Peaks*

Description

Even after calling speaq_align(), the alignment of individual signals is not always perfect, as 'speaq' performs a segment-wise alignment i.e. groups of signals are aligned. For further improvements, partly filled neighboring columns are merged.

Usage

```
combine_peaks(
  shifted_mat = speaq_align(),
  range = 5,
  lower_bound = 1,
  spectrum_data = NULL,
  data_path = NULL
)
```

Arguments

shifted_mat	The matrix returned by speaq_align().
range	Amount of adjacent columns which are permitted to be used for improving the alignment.
lower_bound	Minimum amount of non-zero elements per column to trigger the alignment improvement.
spectrum_data	The list of deconvoluted spectra as returned by generate_lorentz_curves() that was used to generate shifted_mat. No longer required since version 1.2 of Metabodecon.
data_path	If not NULL, the returned dataframes long and short are written to data_path as "aligned_res_long.csv" and "aligned_res_short.csv".

Details

Example of what the function does:

	3.56	3.54	3.51	3.51	3.50
Spectrum 1	0.13	0	0.11	0	0
Spectrum 2	0.13	0	0.12	0	0
Spectrum 3	0.07	0	0	0	0
Spectrum 4	0.08	0	0	0.07	0
Spectrum 5	0.04	0	0.04	0	0

becomes

	3.56	3.54	3.51	3.50
Spectrum 1	0.13	0	0.11	0
Spectrum 2	0.13	0	0.12	0
Spectrum 3	0.07	0	0	0
Spectrum 4	0.08	0	0.07	0
Spectrum 5	0.04	0	0.04	0

I.e. column 3 and 4 get merged, because they are in range of each other and have no common non-zero entries.

Value

A list containing two data frames `long` and `short`. The first data frame contains one one column for each data point in the original spectrum. The second data frame contains only columns where at least one entry is non-zero.

Author(s)

Initial version from Wolfram Gronwald. Refactored by Tobias Schmidt in 2024.

Examples

```
sim_subset <- metabodecon_file("bruker/sim_subset")
spectrum_data <- generate_lorentz_curves_sim(sim_subset)
shifted_mat <- speaq_align(spectrum_data = spectrum_data, verbose = FALSE)
range <- 5
lower_bound <- 1
obj <- combine_peaks(shifted_mat, range, lower_bound)
str(obj)
```

---

convert_pos *Convert from unit A to unit B*

Description

Converts positions/widths from unit A to unit B. If the direction of units A and B is reversed, the width's sign will be reversed as well. To keep widths strictly positive, wrap the result with `abs()`.

Usage

```
convert_pos(xa, ya, yb)

convert_width(xa, ya, yb)
```

Arguments

`xa`	A numeric vector specifying widths/positions in unit A.
`ya, yb`	A numeric vector specifying the positions of at least two points in unit A / unit B.

Value

A numeric vector of values converted from unit A to unit B.

Examples

```
ya <- c(244, 246, 248, 250, 252)
yb <- c(15, 10, 5, 0, -5)
convert_width(c(2, 4, 8), ya, yb)
convert_pos(c(247, 249), ya, yb)
```

`datadir` *Return path to metabodecon's data directory*

Description

Returns the path to the directory where `download_example_datasets()` stores metabodecon's example data sets or any file within that directory. By default this directory is a subdirectory of R's temporary session directory. If `persistent` is set to `TRUE`, the directory equals the data directory returned by `tools::R_user_dir()` instead.

Usage

```
datadir(file = NULL, warn = TRUE, persistent = NULL)
```

Arguments

`file`	Relative path to a file within the data directory.
`warn`	Print a warning message when the requested path does not yet exist?
`persistent`	Return the path to the persistent data directory instead of the temporary one?

Details

The decision to use a temporary data dir as default and a persistent one only optionally was made to conform to CRAN package policies, which state that:

```
Packages should not write in the user's home filespace (including
clipboards), nor anywhere else on the file system apart from the R
session's temporary directory \[...\] Limited exceptions may be allowed
in interactive sessions if the package obtains confirmation from the
user. For R version 4.0 or later \[...\] packages may store user-specific
data, configuration and cache files in their respective user directories
obtained from [tools::R_user_dir()] \[...\].
```

Source: cran.r-project.org/web/packages/policies.

Value

Path to the data directory or a file within it.

See Also

`download_example_datasets()`, `datadir_persistent()`, `datadir_temp()`

Examples

```
# Get temporary datadir and persistent datadir
datadir(persistent = FALSE, warn = FALSE)
datadir(persistent = TRUE,  warn = FALSE)

# Get persistent datadir if existing else temp datadir. Set `warn = TRUE`
# to raise a warning if none of the directories exist yet.
datadir(warn = FALSE)
if (interactive()) datadir()

# Get PERSISTENT_DATADIR/bruker if existing else TEMP_DATADIR/bruker
datadir(file = "bruker/urine", warn = FALSE)
```

---

datadir_persistent *Persistent Data Directory*

Description

Returns the path to the persistent data directory where metabodecon's data sets are stored. This directory equals the data directory returned by `tools::R_user_dir()` plus additional path normalization.

Usage

```
datadir_persistent()
```

Value

Path to the persistent data directory.

See Also

`datadir()`, `datadir_temp()`

Examples

```
datadir_persistent()
```

datadir_temp *Temporary Data Directory*

Description

Returns the path to the temporary data directory where metabodecon's data sets are stored. This directory equals subdirectory 'data' of metabodecons temporary session directory `tmpdir()` plus additional path normalization.

Usage

```
datadir_temp()
```

Value

Returns the path to the temporary data directory.

See Also

`tmpdir()`, `datadir()`, `datadir_persistent()`

Examples

```
datadir_temp()
```

deconvolute *Deconvolute one or more NMR spectra*

Description

Deconvolutes NMR spectra by modeling each detected signal within a spectrum as Lorentz Curve.

Usage

```
deconvolute(
  x,
  nfit = 3,
  smopts = c(2, 5),
  delta = 6.4,
  sfr = NULL,
  wshw = 0,
  ask = FALSE,
  force = FALSE,
  verbose = FALSE,
  nworkers = 1
)
```

```
generate_lorentz_curves(
  data_path,
  file_format = "bruker",
  make_rds = FALSE,
  expno = 10,
  procno = 10,
  raw = TRUE,
  nfit = 10,
  smopts = c(2, 5),
  delta = 6.4,
  sfr = c(11.44494, -1.8828),
  wshw = 0.1527692,
  ask = TRUE,
  force = FALSE,
  verbose = TRUE,
  nworkers = 1
)

generate_lorentz_curves_sim(
  data_path,
  file_format = "bruker",
  make_rds = FALSE,
  expno = 10,
  procno = 10,
  raw = TRUE,
  nfit = 10,
  smopts = c(2, 5),
  delta = 6.4,
  sfr = c(3.55, 3.35),
  wshw = 0,
  ask = FALSE,
  force = FALSE,
  verbose = FALSE,
  nworkers = 1
)
```

Arguments

x	A `spectrum` or `spectra` object as described in metabodecon_classes.
nfit	Integer. Number of iterations for approximating the parameters for the Lorentz curves. See 'Details'.
smopts	Numeric vector with two entries: the number of smoothing iterations and the number of data points to use for smoothing (must be odd). See 'Details'.
delta	Threshold for peak filtering. Higher values result in more peaks being filtered out. A peak is filtered if its score is below $\mu + \sigma \cdot \delta$, where μ is the average peak score in the signal-free region (SFR), and σ is the standard deviation of peak scores in the SFR. See 'Details'.

sfr	Numeric vector with two entries: the ppm positions for the left and right border of the signal-free region of the spectrum. See 'Details'.
wshw	Half-width of the water artifact in ppm. See 'Details'.
ask	Logical. Whether to ask for user input during the deconvolution process. If FALSE, the provided default values will be used.
force	If FALSE, the function stops with an error message if no peaks are found in the signal free region (SFR), as these peaks are required as a reference for peak filtering. If TRUE, the function instead proceeds without peak filtering, potentially increasing runtime and memory usage significantly.
verbose	Logical. Whether to print log messages during the deconvolution process.
nworkers	Number of workers to use for parallel processing. If "auto", the number of workers will be determined automatically. If a number greater than 1, it will be limited to the number of spectra.
data_path	Either the path to a directory containing measured NMR spectra, a dataframe as returned by read_spectrum(), or a list of such dataframes.
file_format	The file_format of the spectrum file. E.g. "bruker" or "jcampdx".
make_rds	Logical or character. If TRUE, stores results as an RDS file on disk. If a character string, saves the RDS file with the specified name. Should be set to TRUE if many spectra are evaluated to decrease computation time.
expno, procno	The experiment/processing number for the file. E.g. "10". Only relevant if file_format equals "bruker". For details see section File Structure in the metabodecon FAQ.
raw	If FALSE, scales the returned signal intensities based on information available in the spectrum metadata, in particular NC_proc. For details see processing-reference.pdf, available at https://www.bruker.com/en.html at section 'Services & Support > Documentation & Manuals > Magnetic Resonance > Acquisition & Processing > TopSpin Processing Commands and Parameters' (requires login).

Details

First, an automated curvature based signal selection is performed. Each signal is represented by 3 data points to allow the determination of initial Lorentz curves. These Lorentz curves are then iteratively adjusted to optimally approximate the measured spectrum.

generate_lorentz_curves_sim() is identical to generate_lorentz_curves() except for the defaults, which are optimized for deconvoluting the 'Sim' dataset, shipped with 'metabodecon'. The 'Sim' dataset is a simulated dataset, which is much smaller than a real NMR spectra and lacks a water signal. This makes it ideal for use in examples or as a default value for functions. However, the default values for sfr, wshw, and delta in the "normal" generate_lorentz_curves() function are not optimal for this dataset. To avoid having to define the optimal parameters repeatedly in examples, this function is provided to deconvolute the "Sim" dataset with suitable parameters.

In generate_lorentz_curves() the parameters nfit, smopts, delta, sfr and wshw must be fully specified. In deconvolute(), these parameters can be set to NULL (the default value). In this case, the function will try to determine the optimal values for these parameters automatically. The values chosen are stored in field args of the returned decon2 object.

Value

A 'decon2' object as described in metabodecon_classes.

Examples

```
## Define the paths to the example datasets we want to deconvolute:
## `sim_dir`: directory containing 16 simulated spectra
## `sim_01`: path to the first spectrum in the `sim` directory
## `sim_01_spec`: the first spectrum in the `sim` directory as a dataframe

sim_dir <- metabodecon_file("sim_subset")
sim_1_dir <- file.path(sim_dir, "sim_01")
sim_2_dir <- file.path(sim_dir, "sim_02")
sim_1_spectrum <- read_spectrum(sim_1_dir)
sim_2_spectrum <- read_spectrum(sim_2_dir)
sim_spectra <- read_spectra(sim_dir)


## Show that `generate_lorentz_curves()` and `generate_lorentz_curves_sim()`
## produce the same results:

sim_1_decon0 <- generate_lorentz_curves(
    data_path = sim_1_dir, # Path to directory containing spectra
    sfr = c(3.55, 3.35), # Borders of signal free region (SFR) in ppm
    wshw = 0, # Half width of water signal (WS) in ppm
    ask = FALSE, # Don't ask for user input
    verbose = FALSE # Suppress status messages
)
sim_1_decon1 <- generate_lorentz_curves_sim(sim_1_dir)
stopifnot(all.equal(sim_1_decon0, sim_1_decon1))


## Show that passing a spectrum produces the same results as passing the
## the corresponding directory:

decon_from_spectrum_dir <- generate_lorentz_curves_sim(sim_1_dir)
decon_from_spectrum_obj <- generate_lorentz_curves_sim(sim_1_spectrum)
decons_from_spectra_obj <- generate_lorentz_curves_sim(sim_spectra)
decons_from_spectra_dir <- generate_lorentz_curves_sim(sim_dir)

most.equal <- function(x1, x2) {
    ignore <- which(names(x1) %in% c("number_of_files", "filename"))
    equal <- all.equal(x1[-ignore], x2[-ignore])
    invisible(stopifnot(isTRUE(equal)))
}

all.equal(  decon_from_spectrum_dir, decon_from_spectrum_obj     )
all.equal(  decons_from_spectra_dir, decons_from_spectra_obj     )
most.equal( decon_from_spectrum_dir, decons_from_spectra_obj[[1]])
most.equal( decon_from_spectrum_dir, decons_from_spectra_dir[[1]])
```

dohCluster *Cluster Based Peak Alignment*

Description

Rewrite of `speaq::dohCluster()`, compatible with the data format returned by 'generate_lorentz_curves()' and 'gen_feat_mat()'. The function name "dohCluster" comes from "Do Hierarchical Clustering" which is part of the Alignment algorithm proposed by Vu et al. (2011) in doi:10.1186/1471-2105-12-405.

Usage

```
dohCluster(
  X,
  peakList,
  refInd = 0,
  maxShift = 100,
  acceptLostPeak = TRUE,
  verbose = TRUE
)
```

Arguments

`X`	Dataframe of signal intensities from all spectra as returned by `gen_feat_mat()`.
`peakList`	List of peak indices as returned `gen_feat_mat()`.
`refInd`	Number of the reference spectrum i.e. the spectrum to which all signals will be aligned to.
`maxShift`	Maximum number of points a value can be moved.
`acceptLostPeak`	Whether to allow the the alignment algorithm to ignore peaks that cannot easily be aligned with the reference spectrum.
`verbose`	Whether to print additional information during the alignment process.

Value

A list containing two data frames `Y` and `new_peakList`. The first one contains the aligned spectra, the second one contains the aligned signals of each spectrum.

Author(s)

Initial version from Wolfram Gronwald. Refactored by Tobias Schmidt in 2024.

Examples

```
sim_subset <- metabodecon_file("bruker/sim_subset")
decons <- generate_lorentz_curves_sim(sim_subset)
feat <- gen_feat_mat(decons)
refObj <- speaq::findRef(feat$peakList)
hclObj <- dohCluster(
     X = feat$data_matrix,
     peakList = feat$peakList,
     refInd = refObj$refInd,
     maxShift = 100,
     acceptLostPeak = TRUE,
     verbose = TRUE
)
str(hclObj, 1)
```

---

download_example_datasets

Download metabodecon Example Datasets

Description

Downloads example datasets that can be used to test the functionality of the metabodecon package. These datasets are not included in the package by default due to size constraints. The datasets are downloaded as zip file and extracted automatically, unless extraction is disabled by the user.

Usage

```
download_example_datasets(
  dst_dir = NULL,
  extract = TRUE,
  persistent = NULL,
  overwrite = FALSE,
  silent = FALSE
)
```

Arguments

dst_dir	The destination directory where the downloaded datasets will be stored. If NULL, the function will return the path to the cached zip file.
extract	Logical. If TRUE, the downloaded zip file will be extracted.
persistent	Logical. If TRUE, the downloaded datasets will be cached at datadir_persistent() to speed up future calls to download_example_datasets(). If FALSE, the datasets will be cached at datadir_temp(). If NULL, the function will check both paths for the cached datasets but will return datadir_temp() if the cached file does not yet exist.
overwrite	Logical. If TRUE, existing files with the same name in the destination directory will be overwritten.
silent	Logical. If TRUE, no output will be printed to the console.

Value

The path to the downloaded (and possibly extracted) datasets.

See Also

datadir()

Examples

```
if (interactive()) {
     zip <- download_example_datasets(extract = FALSE, persistent = FALSE)
     dir <- download_example_datasets(extract = TRUE)
}
```

---

draw_spectrum *Draw Spectrum*

Description

Draws a single spectrum. Internally used by plot_spectrum(), which is usually the recommended way to plot spectra. For usage examples see test/testthat/test-draw_spectrum.R.

Usage

```
draw_spectrum(
  obj,
  foc_rgn = NULL,
  foc_frac = NULL,
  foc_only = TRUE,
  add = FALSE,
  fig_rgn = NULL,
  main = NULL,
  show = TRUE,
  show_d2 = FALSE,
  truepar = NULL,
  mar = c(4.1, 5.1, 0.1, 0.1),
  si_line = list(),
  sm_line = list(),
  sp_line = list(),
  d2_line = list(),
  lc_lines = list(),
  tp_lines = list(),
  cent_pts = list(),
  bord_pts = list(),
  norm_pts = list(),
  bg_rect = list(),
```

```
  foc_rect = list(),
  lc_rects = list(),
  tp_rects = list(),
  bt_axis = list(),
  lt_axis = list(),
  tp_axis = list(),
  rt_axis = list(),
  tp_verts = list(),
  lc_verts = list(),
  lgd = list()
)
```

Arguments

`obj`	An object of type `spectrum` or `decon2`. For details see metabodecon_classes.
`foc_rgn`	Numeric vector specifying the start and end of focus region in ppm.
`foc_frac`	Numeric vector specifying the start and end of focus region as fraction of the full spectrum width.
`foc_only`	Logical. If TRUE, only the focused region is drawn. If FALSE, the full spectrum is drawn.
`add`	If TRUE, draw into the currently open figure. If FALSE, start a new figure.
`fig_rgn`	Drawing region in normalized device coordinates as vector of the form `c(x1, x2, y1, y2)`.
`main`	Main title of the plot. Drawn via `title()`.
`show`	Logical. If FALSE, the function returns without doing anything.
`show_d2`	Logical. If TRUE, the second derivative of the spectrum is drawn. Setting this to TRUE changes most of the defaults for the drawing, e.g. by disabling the drawing of anything related to signal intensities and by changing the y-axis label to "Second Derivative".
`truepar`	Data frame with columns x0, A and lambda containing the true lorentzian that were used to simulate the spectrum. Required if any `tp_*` argument is set.
`mar`	Number of lines below/left-of/above/right-of plot region.
`si_line`, `sm_line`, `lc_lines`, `sp_line`, `d2_line`, `tp_lines`	List of parameters passed to `lines()` when drawing the raw signal intensities, smoothed signal intensities, lorentzian curves found by deconvolution, superposition of lorentzian curves, second derivative and/or true lorentzian curves.
`cent_pts`, `bord_pts`, `norm_pts`	List of parameters passed to `points()` when drawing the peak center points, peak border points and non-peak points.
`bg_rect`, `lc_rects`, `foc_rect`, `tp_rects`	List of parameters passed to `rect()` when drawing the background, lorentzian curve substitutes, focus rectangle and/or true lorentzian curve substitutes.
`bt_axis`, `lt_axis`, `tp_axis`, `rt_axis`	List of parameters used to overwrite the default values passed to `axis()` when drawing the bottom, left, top and right axis. In addition to the parameters of `axis()`, the following additional parameters are supported as well:

- `text`: Description for the axis. Drawn via `mtext()`.
- `n`: Number of tickmarks.
- `digits`: Number of digits for rounding the labels. If a vector of numbers is provided, all numbers are tried, until `n` unique labels are found. See 'Details'.
- `sf`: Scaling factor. Axis values are divided by this number before the labels are calculated. If you set this to anything unequal 1, you should also choose `text` in a way that reflects the scaling. E.g. if you set `sf = 1e6` you could change the text from `"Signal Intensity [au]"` to `"Signal Intensity [Mau]"` or `"Signal Intensity [au] / 1e6"`, with `"Mau"` meaning "Mega-Arbitrary-Units".

`lc_verts, tp_verts`
: List of parameters passed to `abline()` when drawing vertical lines at the centers of estimated lorentzian curves and/or true lorentzian curves. Setting tp_verts$show to TRUE requires `truepar` to be set.

`lgd`
: List of parameters passed to `legend()` when drawing the legend.

Details

Parameters `bt_axis`, `lt_axis`, `tp_axis` and `rt_axis` all support option `n` and `digits`, where `n = 5` means "Draw 5 tickmarks over the full axis range" and `digits = 3` means "round the label shown beside each tickmark to 3 digits". If `n` is omitted, a suitable value is chosen automatically using `axTicks()`. If `digits` is omitted, a default of `2:12` is used. Providing a vector of `digits` causes each digit to be tried as argument for `round()`, until a digit is encountered that results in `n` unique labels. Example:

Assume we have `n = 4` and the corresponding calculated tickmark positions are: 1.02421, 1.02542, 1.02663 and 1.02784. If we provide `digits = 1:5`, the following roundings are tried:

digit	label 1	label 2	label 3	label 4
1	1	1	1	1
2	1.02	1.03	1.03	1.03
3	1.024	1.025	1.027	1.028
4	1.0242	1.0254	1.0266	1.0278
5	1.02421	1.02542	1.02663	1.02784

In the above example the process would stop at `digit = 3`, because at this point we have `n = 4` unique labels (1.024, 1.025, 1.027 and 1.028).

Value

NULL. Called for side effect of plotting.

Examples

```
decon <- deconvolute(sim[[1]], sfr = c(3.55, 3.35))
draw_spectrum(obj = decon)
draw_spectrum(obj = decon, lgd = list(x = "top", bg = NA))
```

```
draw_spectrum(obj = decon, foc_rgn = c(3.45, 3.37))
draw_spectrum(obj = decon, fig = c(0.1, 0.4, 0.30, 0.45), add = TRUE)
draw_spectrum(obj = decon, fig = c(0.1, 0.4, 0.05, 0.20), add = FALSE)
draw_spectrum(obj = decon, lc_lines = NULL, lc_rects = NULL, foc_only = FALSE)
```

`evalwith` *Evaluate an expression with predefined global state*

Description

Evaluates an expression with a predefined global state, including the:

- working directory (set via `setwd()`)
- global options (set via `options()`)
- graphical parameters (set via `par()`)

In addition to that, `evalwith` allows to:

- Redirect or capture the output and/or message stream via `sink()`
- Measure the runtime of the evaluated expression via `system.time()`
- Creating a temporary test directory (inside `tmpdir()`) and populating it with input files according to `inputs`
- Predefine answers for calls to `readline()` happening during evaluation of `expr`
- Caching the result of the expression

All changes to the global state are reverted after the expression has been evaluated.

Usage

```
evalwith(
  expr,
  testdir = NULL,
  answers = NULL,
  output = NULL,
  message = NULL,
  plot = NULL,
  datadir_temp = c("default", "missing", "empty", "filled")[1],
  datadir_persistent = c("default", "missing", "empty", "filled")[1],
  inputs = character(),
  opts = NULL,
  pars = NULL,
  cache = FALSE,
  overwrite = FALSE
)
```

Arguments

expr	Expression to be evaluated.
testdir	ID of the test directory. E.g. "xyz/2". Will be created and populated with inputs. To clear, use clear(testdir("xyz/2")).
answers	Answers to be returned by readline().
output	Path to the file where output stream should be redirected to. Use "captured" to capture the output.
message	Path to the file where message stream be redirected to. Use "captured" to capture the messages.
plot	An expression opening a device, the string "captured" or a path ending in ".pdf", ".svg", or ".png". Examples: svg("tmp.svg"), quote(pdf("tmp.pdf")), "captured", "tmp.png". Passing "captured" is equivalent to passing tempfile(fileext = ".png").
datadir_temp	State of the mocked temporary data directory. See details section.
datadir_persistent	State of the mocked persistent data directory. See details section.
inputs	Paths to be copied to the test directory before evaluating expr.
opts	Named list of options to be set. See options().
pars	Named list of parameters to be set. See par().
cache	Logical indicating whether to cache the result of the expression.
overwrite	Logical indicating whether to overwrite the cache file if it already exists.

Details

The datadir_temp and datadir_persistent arguments accept values "missing", "filled" and "empty". Setting a value unequal NULL causes the functions datadir_temp() and/or datadir_persistent() to be replaced with mock functions pointing to fake directories. Functions depending on these functions will then use the fake directories instead of the real ones. When set to "missing" the returned mock directory does not exist. When set to "empty" it exists and is guaranteed to be empty. When set to "filled", it is populated with example datasets.

Attention: the mocked functions, i.e. datadir_temp() and datadir_persistent() cannot be used directly inside expr when called via devtools::test(). I'm not sure why, but it seems as if devtools and/or testthat have their own copies of the functions which are used when the expression is evaluated.

Value

A list containing with following elements:

- rv: The return value of the expression.
- runtime: The "elapsed" runtime of the expression in seconds. Measured with system.time().
- output: The captured output.
- message: The captured messages.
- plot: The path to the saved plot.
- testdir: The path to the test directory.
- inputs: The paths to the copied input files.

Examples

```
x1 <- evalwith(output = "captured", cat("Helloworld\n"))
str(x1)

x2 <- evalwith(datadir_persistent = "missing", message = "captured", datadir())
str(x2)

x3 <- evalwith(testdir = "dummy", inputs = "bruker/urine/urine_1", dir())
str(x3)

x4 <- evalwith(Sys.sleep(0.02))
str(x4)
```

---

gen_feat_mat *Generate Feature Matrix.*

Description

Generate a feature matrix.

Usage

```
gen_feat_mat(
  data_path,
  ppm_range = get_ppm_range(data_path),
  si_size_real_spectrum = length(data_path$y_values),
  scale_factor_x = 1000,
  warn = TRUE
)
```

Arguments

data_path	A list of deconvoluted spectra as returned by `generate_lorentz_curves()`. In older versions, this could also be the path passed to `generate_lorentz_curves()`, but this is deprecated and will trigger a warning. See 'Details' for more information.
ppm_range	The ppm range over which your signals are distributed.
si_size_real_spectrum	Number of data points in your spectra.
scale_factor_x	The x scale factor used during the deconvolution.
warn	Whether to print a warning in case a file path is passed to `data_path` instead of a list of deconvoluted spectra.

Details

Before version 1.2 of 'metabodecon', the deconvolution functions `generate_lorentz_curves` and `MetaboDecon1D` wrote their output partially as txt files to their input folder. Back then, `gen_feat_mat()` used those txt files as input to generate the feature matrix. Since version 1.2 these txt files are no longer created by default, to prevent accidental modifications of the input folders. Therefore, the recommended way to pass the required information to `gen_feat_mat()` is to directly pass the output of `generate_lorentz_curves()` to `gen_feat_mat()`. However, to stay backwards compatible, the name of parameter `data_path` was not changed and passing an actual path to `data_path` is still possible, but will result in a warning (unless `warn` is set to `FALSE`).

Value

A list with the following elements:

`data_matrix`: A data.frame where each row corresponds to one spectrum and each column to one data point, i.e. for 10 input spectra with 131072 data points each `data_matrix` would have dimensions 10 x 131072.

`peakList`: A list of vectors, where each vector contains the indices of the peaks in the corresponding spectrum. The indices increase from left to right, i.e. the smallest index corresponds to the highest ppm value, as the ppm values decrease from left to right.

`w`: A list of vectors where each vector contains the "position parameter" of the peaks in the corresponding spectrum.

`A`: A list of vectors where each vector contains the "area parameter" of the peaks in the corresponding spectrum.

`lambda`: A list of vectors where each vector contains the "width parameter" of the peaks in the corresponding spectrum.

Author(s)

Initial version from Wolfram Gronwald. Refactored by Tobias Schmidt in 2024.

Examples

```
sim_subset <- metabodecon_file("sim_subset")
decons <- generate_lorentz_curves_sim(sim_subset)
obj <- gen_feat_mat(decons)
str(obj, 2, give.attr = FALSE)
```

---

get_data_dir *Retrieve directory path of an example dataset*

Description

Returns the path to the directory storing the example files shipped with metabodecon.

Deprecated since metabodecon v1.2.0. Please use `datadir()` instead. See examples below for usage.

[Deprecated]

Usage

```
get_data_dir(dataset_name = c("", "blood", "test", "urine"), warn = TRUE)
```

Arguments

dataset_name	Either "", "test", "blood" or "urine".
warn	Whether to print a warning message when the example folders do not yet exist, i.e. download_example_datasets() has not been called yet.

Value

Path to the directory storing the example files.

See Also

download_example_datasets()

Examples

```
x <- get_data_dir("urine")                    # Deprecated
y <- datadir("example_datasets/bruker/urine")  # Preferred
cat(x, y, sep = "\n")
```

---

get_ppm_range *Get PPM Range covered by Spectra*

Description

Returns the ppm range across all peaks of the provided deconvoluted spectra.

Usage

```
get_ppm_range(spectrum_data, full_range = FALSE)
```

Arguments

spectrum_data	A list of deconvoluted spectra as returned by generate_lorentz_curves().
full_range	If TRUE, the full range of the spectra is returned. If FALSE, only the range from the lowest to the highest peak center is returned.

Value

A vector containing the lowest and highest ppm value over all peaks of the provided deconvoluted spectra.

Author(s)

Wolfram Gronwald, 2023: initial version. Tobias Schmidt, 2024: .

Examples

```
spectrum_data <- generate_lorentz_curves(
    data_path = sim[1:2],
    nfit = 3,
    sfr = c(3.55, 3.35),
    wshw = 0,
    ask = FALSE,
    verbose = FALSE
)
ppm_rng <- get_ppm_range(spectrum_data)
print(ppm_rng)
```

`is_metabodecon_class` *Is an Object from a Metabodecon Class?*

Description

Check if an object is an instance of a specific 'Metabodecon Class'. See metabodecon_classes for a list of classes.

Usage

```
is_spectrum(x, check_class = TRUE, check_contents = FALSE)

is_ispec(x)

is_idecon(x)

is_decon0(x)

is_decon1(x)

is_decon2(x)

is_align(x)

is_spectra(
  x,
  check_class = TRUE,
  check_contents = FALSE,
  check_child_classes = FALSE
)

is_ispecs(x)

is_idecons(x)
```

```
is_decons0(x)

is_decons1(x)

is_decons2(x)

is_aligns(x)
```

Arguments

x	The object to check.
check_class	Logical indicating whether to check the class of the object.
check_contents	Logical indicating whether to check the contents of the object.
check_child_classes	Logical indicating whether to check the class of each element of the object.

Value

TRUE if the object is an instance of the specified class, otherwise FALSE.

Examples

```
ss <- sim[1:2]
dd <- deconvolute(ss, sfr = c(3.55, 3.35))
aa <- align(dd)
s1 <- sim[[1]]
d1 <- dd[[1]]
a1 <- aa[[1]]

is_spectrum(s1) # TRUE
is_spectrum(s1, check_contents = TRUE) # TRUE
is_decon0(d1) # FALSE
is_decon1(d1) # FALSE
is_decon2(d1) # TRUE
is_align(a1)  # TRUE

is_spectra(ss) # TRUE
is_decons0(dd) # FALSE
is_decons1(dd) # FALSE
is_decons2(dd) # TRUE
is_aligns(aa)  # TRUE
```

---

make_spectrum *Create a Spectrum Object*

Description

Creates a spectrum object from the provided signal intensities, frequencies and chemical shifts.

Usage

```
make_spectrum(
  si,
  cs_max,
  cs_width,
  fq_ref,
  fq_width = NULL,
  force = FALSE,
  silent = FALSE,
  name = NULL,
  path = NULL,
  type = NULL,
  simpar = NULL,
  mfs = NULL
)
```

Arguments

si	Numeric vector of signal intensities, ordered from highest to lowest corresponding chemical shift.
cs_max	The highest chemical shift value in ppm, usually shown as left end of the spectrum.
cs_width	The width of the spectrum in ppm.
fq_ref	The reference frequency in Hz.
fq_width	The width of the spectrum in Hz. Only used to check whether the values calculated from cs_max, cs_width and fq_ref match the provided value. If NULL, this check will be skipped.
force	If TRUE, the function will not raise an error in case of discrepancies between the calculated and the provided spectrum width in Hz, but will print a info message instead. To hide this message as well, set silent = TRUE.
silent	If TRUE, no output will be printed to the console.
name	The name of the spectrum, e.g. "Blood 1" or "Urine Mouse X23D".
path	The path to the spectrum file, e.g. "/example_datasets/bruker/urine/urine_1".
type	The type of experiment, e.g. "H1 CPMG" or "H1 NOESY".
simpar	The simulation parameters used to generate the spectrum.
mfs	The magnetic field strength in Tesla.

Value

A spectrum object as described in metabodecon_classes.

Examples

```
si <- c(1, 1, 3, 7, 8, 3, 8, 5, 2, 1)
cs_max <- 14.8
```

```
cs_width <- 20.0
fq_ref <- 600.25 * 1e6
fq_width <- 12005
spectrum <- make_spectrum(si, cs_max, cs_width, fq_ref, fq_width)
spectrum2 <- make_spectrum(si, cs_max, cs_width, fq_ref, fq_width = 12010, force = FALSE)
```

`MetaboDecon1D` *Deconvolute 1D NMR spectrum*

Description

Automatic deconvolution of a 1D NMR spectrum into several Lorentz curves and the integration of them. The NMR file needs to be in Bruker format or jcamp-dx format.

Deprecated since metabodecon v1.2.0. Please use `generate_lorentz_curves()` instead. See examples below for usage.

[Deprecated]

Usage

```
MetaboDecon1D(
  filepath,
  filename = NA,
  file_format = "bruker",
  number_iterations = 10,
  range_water_signal_ppm = 0.1527692,
  signal_free_region = c(11.44494, -1.8828),
  smoothing_param = c(2, 5),
  delta = 6.4,
  scale_factor = c(1000, 1e+06),
  debug = FALSE,
  store_results = NULL
)
```

Arguments

`filepath`	Complete path of the file folder (Notice for Bruker format: filepath needs to be the spectrum folder containing one or more different spectra (e.g."C:/Users/Username/Desktop/spectra_fr
`filename`	Name of the NMR file. (Notice for Bruker format: filename need to be the name of your spectrum which is also the name of the folder) (Default: filename = NA to analyze more spectra at once)
`file_format`	Format (bruker or jcampdx) of the NMR file. (Default: file_format = "bruker")
`number_iterations`	Number of iterations for the approximation of the parameters for the Lorentz curves (Default: number_iterations=10)
`range_water_signal_ppm`	Half width of the water artefact in ppm (Default: range_water_signal=0.1527692 (e.g. for urine NMR spectra))

`signal_free_region`
: Row vector with two entries consisting of the ppm positions for the left and right border of the signal free region of the spectrum. (Default: signal_free_region=c(11.44494, -1.8828))

`smoothing_param`
: Row vector with two entries consisting of the number of smoothing repeats for the whole spectrum and the number of data points (uneven) for the mean calculation (Default: smoothing_param=c(2,5))

`delta`
: Defines the threshold value to distinguish between signal and noise (Default: delta=6.4)

`scale_factor`
: Row vector with two entries consisting of the factor to scale the x-axis and the factor to scale the y-axis (Default: scale_factor=c(1000,1000000))

`debug`
: Logical value to activate the debug mode (Default: debug=FALSE)

`store_results`
: Specifies whether the lorentz curve parameters `A`, `lambda` and `x_0` and the approximated spectrum should be stored on disk (in addition to returning them). If `store_results` is `NULL` (default), the user is asked interactively where the files should be stored. If FALSE, the results are not stored. If TRUE, the results are stored in a subdirectory of R's per-session temporary directory.

Value

A `decon0` object as described in metabodecon_classes.

Author(s)

Martina Haeckl, 2021: Initial version. Tobias Schmidt, 2024: Added parameters debug and `store_results`. Added minor improvements to pass CRAN checks.

References

Haeckl, M.; Tauber, P.; Schweda, F.; Zacharias, H.U.; Altenbuchinger, M.; Oefner, P.J.; Gronwald, W. An R-Package for the Deconvolution and Integration of 1D NMR Data: MetaboDecon1D. Metabolites 2021, 11, 452. https://doi.org/10.3390/metabo11070452

See Also

`calculate_lorentz_curves()`, `plot_triplets()`, `plot_lorentz_curves_save_as_png()`, `plot_spectrum_superpos`

Examples

```
## ATTENTION: using MetaboDecon1D() for deconvolution is deprecated. Please use
## generate_lorentz_curves() instead.

## The following example shows how a subset of the Sim dataset, consisting
## of two spectrum objects, can be deconvoluted using `MetaboDecon1D()`. The
## whole example code is wrapped into `evalwith()` to simulate user input.
## When using the function interactively, you should type in the answers to
## the questions manually.
expected_answers <- c(
```

```
      "10",   # Subfolder of your filepath, i.e. the experiment number?
      "10",   # Subsubsubfolder of filepath, i.e. the processing number?
      "y",    # Use same parameters for all spectra?
      "1",    # File to adjust all parameters.
      "n",    # Signal free region borders correct selected?
      "3.55", # Left border.
      "3.35", # Right border.
      "y",    # Signal free region borders correct selected?
      "n",    # Water artefact fully inside red vertical lines?
      "0",    # Half width range (in ppm) for the water artefact.
      "y",    # Water artefact fully inside red vertical lines?
      "n"     # Save results as text documents?
)
sim <- metabodecon_file("bruker/sim_subset")
evalwith(answers = expected_answers, {
     sim_decon <- MetaboDecon1D(sim)
})

## Deconvolute only the first spectrum of the folder "bruker/sim_subset" into
evalwith(answers = expected_answers[-(3:4)], {
     sim_decon <- MetaboDecon1D(sim, filename = "sim_01")
})
```

`metabodecon_classes` *Metabodecon Classes*

Description

Metabodecon introduces a set of classes to highlight the presence of certain elements in corresponding objects.

The order of elements may vary between different versions of Metabodecon, thus elements should always be accessed by name, for example, using `x$si` or `x[["cs"]]`. A short description of each class is given in the listing below.

- `spectrum`: One NMR spectrum
- `decon0`: One deconvoluted NMR spectrum stored in `MetaboDecon1D()` format
- `decon1`: One deconvoluted NMR spectrum stored in `generate_lorentz_curves()` format
- `decon2`: One deconvoluted NMR spectrum stored in `deconvolute()` format
- `align`: One aligned NMR spectrum

The classes mentioned above represent individual objects, such as a single spectrum, deconvolution, or alignment. However, it is often useful to describe collections of these objects, such as a list of spectra or deconvolutions. Therefore, for each individual class, a corresponding "collection" class is provided. These collection classes are named: `spectra`, `decons0`, `decons1`, `decons2`, and `aligns`.

More details can be found in Metabodecon's online documentation at spang-lab.github.io/metabodecon/articles/Metabodecon Classes.

metabodecon_file *Return Path to File or Directory in metabodecon Package*

Description

Recursively searches for files or directories within the 'metabodecon' package that match the given name.

Usage

```
metabodecon_file(name = "sim_01")
```

Arguments

name	The name to search for.

Value

The file or directory path.

Examples

```
# Unambiguous paths
metabodecon_file("urine_1")
metabodecon_file("urine_1.dx")
metabodecon_file("sim/sim_01")

# Ambiguous paths (i.e. multiple matches)
metabodecon_file("sim")
metabodecon_file("urine")

# Non-existing paths (i.e. a character vector of length zero gets returned)
metabodecon_file("asdfasdf")
```

plot_lorentz_curves_save_as_png *Plot lorentz curves for variable range*

Description

Plots the original spectrum and all generated Lorentz curves and save the result as png under the filepath.

Superseded by `plot_spectrum()` since metabodecon v1.2.0. Will be replaced with v2.**[Deprecated]**

Usage

```
plot_lorentz_curves_save_as_png(
  deconv_result,
  x_range = c(),
  y_range = c(),
  out_dir = ".",
  ask = TRUE
)
```

Arguments

deconv_result	Saved result of the MetaboDecon1D() function
x_range	Row vector with two entries consisting of the ppm start and the ppm end value to scale the range of the x-axis (optional)
y_range	Row vector with two entries consisting of the ppm start and the ppm end value to scale the range of the y-axis (optional)
out_dir	Path to the directory where the png files should be saved. Default is the current working directory.
ask	Logical value. Whether to ask for confirmation from the user before writing files to disk. Default is TRUE.

Value

NULL, called for side effects.

See Also

MetaboDecon1D(), plot_triplets(), plot_spectrum_superposition_save_as_png()

Examples

```
sim <- metabodecon_file("bruker/sim_subset")
sim_decon <- generate_lorentz_curves_sim(sim)
png_dir <- tmpdir("sim_decon/pngs", create = TRUE)
plot_lorentz_curves_save_as_png(sim_decon, out_dir = png_dir, ask = FALSE)
dir(png_dir, full.names = TRUE)
```

---

plot_spectra *Plot Spectra*

Description

Plot a set of deconvoluted spectra.

Usage

```
plot_spectra(
  obj,
  ...,
  sfy = 1e+06,
  xlab = "Chemical Shift [ppm]",
  ylab = paste("Signal Intensity [au] /", sfy),
  mar = c(4.1, 4.1, 1.1, 0.1)
)
```

Arguments

obj	An object of type decons0, decons1 or decons2. For details see metabodecon_classes.
...	Additional arguments passed to the conversion function.
sfy	Scaling factor for the y-axis.
xlab	Label for the x-axis.
ylab	Label for the y-axis.
mar	A numeric vector of length 4, which specifies the margins of the plot.

Value

A plot of the deconvoluted spectra.

See Also

plot_spectrum() for a much more sophisticated plotting routine suitable for plotting a single spectrum.

Examples

```
obj <- deconvolute(sim[1:4], sfr = c(3.55, 3.35))
plot_spectra(obj)
```

---

plot_spectrum *Plot Spectrum*

Description

Plot a spectrum and zoom in on a specific region. **[Experimental]**

Usage

```
plot_spectrum(
  x,
  ...,
  obj = as_v2_obj(x),
  foc_frac = get_foc_frac(obj),
  foc_rgn = get_foc_rgn(obj, foc_frac),
  sub1 = TRUE,
  sub2 = FALSE,
  sub3 = width(foc_rgn) < width(obj$cs),
  mar = NULL,
  frame = FALSE,
  con_lines = TRUE
)
```

Arguments

x	An object of type spectrum, decon0, decon1 or decon2. For details see metabodecon_classes.
...	Additional arguments passed to draw_spectrum() for **every** sub figure. See 'Details'.
obj	An object of type spectrum or decon2. Usually auto generated from x, but can be set manually in case the default conversion is not sufficient.
foc_frac	A numeric vector specifying the start and end of the focus region as fraction of the full spectrum width. Only used if foc_rgn is set to NULL.
foc_rgn	A numeric vector specifying the start and end of the focus region in ppm. If set to NULL, foc_frac is used to determine the focus region. If both foc_rgn and are set to NULL, a suitable focus region is chosen automatically. Takes precedence over foc_frac.
sub1, sub2, sub3	List of arguments passed to draw_spectrum() when drawing sub figure 1-3. See 'Details'.
mar	A numeric vector of length 4 passed, which specifies the margins of the plot. Passed to par(). If set to NULL, a suitable value is chosen automatically.
frame	A list of values passed to box() when drawing the frame around plot region. If set to NULL, no frame is drawn.
con_lines	A list of values passed to lines() when drawing the connecting lines between sub figure 1 and the focus rectangle in sub figure 3. See 'Details'. If set to NULL, the connecting lines are not drawn.

Details

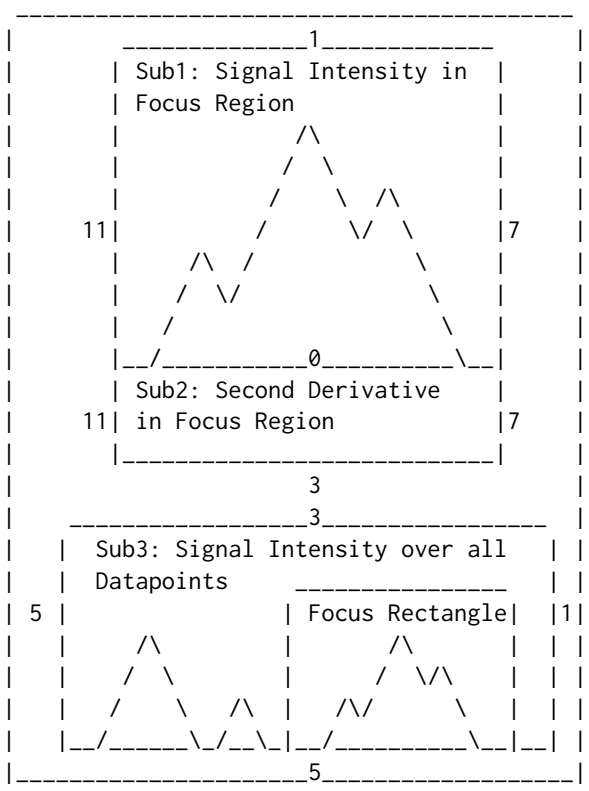
This function first initializes a new plotting canvas. After that it calls draw_spectrum() multiple times to draw the following sub figures onto the plotting canvas:

1. The signal intensities in the focus region
2. The second derivative in the focus region

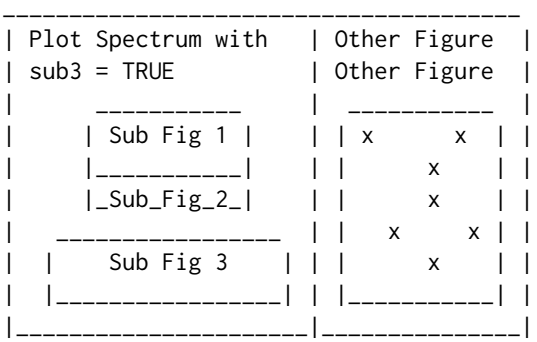
3. The signal intensities over all datapoints

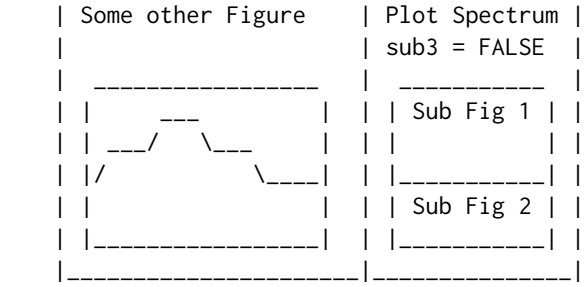
The argument lists for the individual calls to `draw_spectrum()` are determined at runtime and depend on the arguments passed to `plot_spectrum()` as well as the currently active graphics device. To customize the appearance of the individual sub plots, you can overwrite each value passed to `draw_spectrum()` by providing a corresponding named element in `sub1`, `sub2` or `sub3`.

A sketch of the resulting figure is shown below.



Note that the figure created by `plot_spectrum()` can be part of a multi-figure configuration as created when setting `mfrow` or `mfcol` via `par()`. Example:





Value

NULL. Called for side effect of plotting as sketched in 'Details'.

Examples

```
## 1. Prepare a deconvoluted spectrum as input

spec <- read_spectrum(metabodecon_file("sim/sim_01"))
decon <- generate_lorentz_curves_sim(spec)

## 2.1. Plot the full (non-deconvoluted) spectrum
## 2.2. Remove connecting lines, and focus on a specific region specified in ppm
## 2.3. Show second derivative and focus on a specific region specified as fraction
## 2.4. Change color of focus rectangle and margins of sub figure 1
## 2.5. Hide xlab and show second derivative
## 2.6. Change the figure region for sub figure 1

plot_spectrum(spec, sub1 = FALSE)
plot_spectrum(decon, foc_rgn = c(3.49, 3.45), con_lines = FALSE)
plot_spectrum(decon, sub2 = TRUE, foc_frac = c(0.40, 0.30))
plot_spectrum(decon,
    sub1 = list(mar = c(3, 6, 3, 6), lt_axis = list(col = "violet")),
    foc_rect = list(border = "violet", col = transp("violet")),
    con_lines = list(col = "violet")
)
plot_spectrum(decon,
    sub2 = TRUE,
    sub3 = list(bt_axis = list(text = "")),
    frame = TRUE,
    con_lines = FALSE
)
plot_spectrum(decon, sub1 = list(fig_rgn_npc = c(0,1,.3,1), mar = c(0,5,0,0)))
```

`plot_spectrum_superposition_save_as_png` *Plot spectrum approx for variable range*

Description

Plots the original spectrum and the superposition of all generated Lorentz curves and saves the result as png under the specified filepath.

Superseded by `plot_spectrum()` since metabodecon v1.2.0. Will be replaced with v2.

[Deprecated]

Usage

```
plot_spectrum_superposition_save_as_png(
  deconv_result,
  x_range = c(),
  y_range = c(),
  out_dir = ".",
  ask = TRUE
)
```

Arguments

`deconv_result`	Saved result of the MetaboDecon1D() function
`x_range`	Row vector with two entries consisting of the ppm start and the ppm end value to scale the range of the x-axis (optional)
`y_range`	Row vector with two entries consisting of the ppm start and the ppm end value to scale the range of the y-axis (optional)
`out_dir`	Path to the directory where the png files should be saved. Default is the current working directory.
`ask`	Logical value. Whether to ask for confirmation from the user before writing files to disk.

Value

NULL, called for side effects.

Author(s)

Martina Haeckl, 2021.

See Also

`MetaboDecon1D()`, `calculate_lorentz_curves()`, `plot_triplets()`, `plot_lorentz_curves_save_as_png()`

Examples

```
sim <- metabodecon_file("bruker/sim_subset")
sim_decon <- generate_lorentz_curves_sim(sim)
png_dir <- tmpdir("sim_decon/pngs", create = TRUE)
plot_spectrum_superposition_save_as_png(sim_decon, out_dir = png_dir, ask = FALSE)
dir(png_dir, full.names = TRUE)
```

plot_triplets *Plot peak triplets for variable range*

Description

Plots the peak triplets for each peak detected by `MetaboDecon1D()` and stores the plots as png at `outdir`.

Superseded by `plot_spectrum()` since metabodecon v1.2.0. Will be replaced with v2.

[Deprecated]

Usage

```
plot_triplets(
  deconv_result,
  x_range = c(),
  y_range = c(),
  out_dir = ".",
  ask = TRUE
)
```

Arguments

`deconv_result`	Saved result of the MetaboDecon1D() function
`x_range`	Row vector with two entries consisting of the ppm start and the ppm end value to scale the range of the x-axis (optional)
`y_range`	Row vector with two entries consisting of the ppm start and the ppm end value to scale the range of the y-axis (optional)
`out_dir`	Directory to save the png files (optional)
`ask`	Logical value to ask the user if the png files should be saved in the specified directory (optional)

Value

No return value, called for side effect of plotting.

Author(s)

Martina Haeckl, 2021: Initial version. Tobias Schmidt, 2024: Minor updates to pass CRAN checks.

See Also

`MetaboDecon1D()`, `calculate_lorentz_curves()`, `plot_lorentz_curves_save_as_png()`, `plot_spectrum_superpos`

Examples

```
sim <- metabodecon_file("bruker/sim_subset")
sim_decon <- generate_lorentz_curves_sim(sim)
png_dir <- tmpdir("sim_decon/pngs", create = TRUE)
plot_triplets(sim_decon, out_dir = png_dir, ask = FALSE)
dir(png_dir, full.names = TRUE)
```

`print_methods` *S3 Methods for Printing Metabodecon Objects*

Description

S3 Methods for printing metabodecon objects as described in the Metabodecon Classes.

Usage

```
## S3 method for class 'spectrum'
print(x, name = FALSE, ...)

## S3 method for class 'ispec'
print(x, name = FALSE, ...)

## S3 method for class 'idecon'
print(x, name = FALSE, ...)

## S3 method for class 'decon1'
print(x, name = FALSE, ...)

## S3 method for class 'decon2'
print(x, name = FALSE, ...)

## S3 method for class 'align'
print(x, name = FALSE, ...)

## S3 method for class 'spectra'
print(x, ...)

## S3 method for class 'ispecs'
print(x, ...)

## S3 method for class 'idecons'
print(x, ...)

## S3 method for class 'decons1'
print(x, ...)

## S3 method for class 'decons2'
```

```
print(x, ...)

## S3 method for class 'aligns'
print(x, ...)
```

Arguments

x	The object to print.
name	Logical. If TRUE, the name of the object is printed before the object.
...	Not used. Only accepted to comply with generic base::print().

Value

NULL, called for side effect of printing to the standard output device.

Examples

```
si <- c(1, 1, 3, 7, 8, 3, 8, 5, 2, 1)
cs_max <- 14.8
cs_width <- 20.0
fq_ref <- 600.25 * 1e6
fq_width <- 12005
spectrum <- read_spectrum()
print(spectrum)
```

---

read_spectra *Read one or more spectra from Disk*

Description

read_spectrum() reads a single spectrum from disk and returns it as spectrum object. read_spectra() can be used to read multiple spectra at once and returns a spectra object.

Usage

```
read_spectra(
  data_path = pkg_file("example_datasets/bruker/urine"),
  file_format = "bruker",
  expno = 10,
  procno = 10,
  raw = FALSE,
  silent = TRUE,
  force = FALSE
)
```

Arguments

data_path	The path of the file/folder containing the spectrum data. E.g. "example_datasets/jcampdx/urine/urir or "example_datasets/bruker/urine/urine".
file_format	The file_format of the spectrum file. E.g. "bruker" or "jcampdx".
expno, procno	The experiment/processing number for the file. E.g. "10". Only relevant if file_format equals "bruker". For details see section File Structure in the metabodecon FAQ.
raw	If FALSE, scales the returned signal intensities based on information available in the spectrum metadata, in particular NC_proc. For details see processing-reference.pdf, available at https://www.bruker.com/en.html at section 'Services & Support > Documentation & Manuals > Magnetic Resonance > Acquisition & Processing > TopSpin Processing Commands and Parameters' (requires login).
silent	If TRUE, no output will be printed to the console.
force	If TRUE, try to continue when encountering errors and print info messages instead. To hide these messages as well, set silent = TRUE.

Value

A spectrum object as described in metabodecon_classes.

Examples

```
relpath <- "example_datasets/bruker/urine"
urine <- system.file(relpath, package = "metabodecon")
urine_1 <- file.path(urine, "urine_1")
urine_2 <- file.path(urine, "urine_2")
x1 <- read_spectrum(urine_1)
x2 <- read_spectrum(urine_2)
xx <- read_spectra(urine)
str(xx)
str(x1)
stopifnot(all.equal(x1, xx$urine_1))
```

---

read_spectrum *Read one or more spectra from Disk*

Description

read_spectrum() reads a single spectrum from disk and returns it as spectrum object. read_spectra() can be used to read multiple spectra at once and returns a spectra object.

Usage

```
read_spectrum(
  data_path = metabodecon_file("bruker/sim/sim_01"),
  file_format = "bruker",
  expno = 10,
  procno = 10,
  raw = FALSE,
  silent = TRUE,
  force = FALSE
)
```

Arguments

data_path	The path of the file/folder containing the spectrum data. E.g. "example_datasets/jcampdx/urine/urir or "example_datasets/bruker/urine/urine".
file_format	The file_format of the spectrum file. E.g. "bruker" or "jcampdx".
expno, procno	The experiment/processing number for the file. E.g. "10". Only relevant if file_format equals "bruker". For details see section File Structure in the metabodecon FAQ.
raw	If FALSE, scales the returned signal intensities based on information available in the spectrum metadata, in particular NC_proc. For details see processing-reference.pdf, available at https://www.bruker.com/en.html at section 'Services & Support > Documentation & Manuals > Magnetic Resonance > Acquisition & Processing > TopSpin Processing Commands and Parameters' (requires login).
silent	If TRUE, no output will be printed to the console.
force	If TRUE, try to continue when encountering errors and print info messages instead. To hide these messages as well, set silent = TRUE.

Value

A spectrum object as described in metabodecon_classes.

Examples

```
relpath <- "example_datasets/bruker/urine"
urine <- system.file(relpath, package = "metabodecon")
urine_1 <- file.path(urine, "urine_1")
urine_2 <- file.path(urine, "urine_2")
x1 <- read_spectrum(urine_1)
x2 <- read_spectrum(urine_2)
xx <- read_spectra(urine)
str(xx)
str(x1)
stopifnot(all.equal(x1, xx$urine_1))
```

sap *The SAP Dataset*

Description

The SAP Dataset consists of a single 'Simple-As-Possible' (SAP) spectrum. The purpose of the SAP spectrum is to provide a straightforward example that can be used to test and understand the deconvolution algorithm in detail.

Usage

```
sap
```

Format

An object of class `spectra` of length 1.

Details

The first (and only) spectrum within the SAP dataset contains 128 datapoints ranging from -6.4 to 6.4 ppm with four peaks. A rough sketch of the spectrum is shown below:

```
-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~
|      SFR      |              w              |     SFR      |
|               |  x          www       p     |              |
|~-~-~-~-~-~-~-~|~-|-|-~-~-~-~-~|~-~-~-~-|-~-~-~-~-~-~-~-~-~-~-~
|               |  | |          |         |      |
6.4             |  | 2.24       0.047     -2.22  -3.2
                |  2.61
                3.2
```

sim *The Sim Dataset*

Description

A simulated dataset generated from the Blood dataset.

Usage

```
sim
```

Format

A `spectra` object consisting of 16 `spectrum` objects, where each spectrum contains 2048 datapoints ranging from 3.60 to 3.29 ppm. For details about `spectrum` and `spectra` objects see metabodecon_classes.

`simulate_spectrum` *Simulate a 1D NMR Spectrum*

Description

Simulates a 1D NMR spectrum based on the provided parameters. **[Experimental]**

Usage

```
simulate_spectrum(
  name = "sim_00",
  seed = sum(utf8ToInt(name)),
  ndp = 2048,
  npk = 10,
  csres = 0.00015,
  cs = seq(from = 3.6, length.out = ndp, by = -csres),
  pkr = quantile(cs, c(0.25, 0.75)),
  fqref = 600252806.95,
  x0 = sort(runif(npk, pkr[1], pkr[2])),
  A = runif(npk, 2.5, 20) * 1000,
  lambda = runif(npk, 0.9, 1.3)/1000,
  noise = rnorm(length(cs), sd = 1200)
)
```

Arguments

`name`	The name of the spectrum.
`seed`	The seed for the random number generator.
`ndp`	The number of data points in the spectrum.
`npk`	The number of peaks in the spectrum.
`csres`	The chemical shift resolution in PPM.
`cs`	The vector of chemical shifts in PPM.
`pkr`	The start and stop of the peak region in PPM.
`fqref`	The reference frequency in Hz.
`x0`	The peak center positions in PPM.
`A`	The peak area parameter.
`lambda`	The peak width parameter.
`noise`	The noise to add to the spectrum.

Value

A `spectrum` object as described in metabodecon_classes.

Examples

```
simA <- simulate_spectrum("simA")
simA_copy <- simulate_spectrum("simA")
simB <- simulate_spectrum("simB")
simC <- simulate_spectrum("simC", npk = 20)
plot_spectrum(simC)
if (!identical(simA, simA_copy)) stop()
if ( identical(simA, simB     )) stop()
```

`speaq_align` *Align Signals using 'speaq'*

Description

Performs signal alignment across the individual spectra using the 'speaq' package (Beirnaert C, Meysman P, Vu TN, Hermans N, Apers S, Pieters L, et al. (2018) speaq 2.0: A complete workflow for high-throughput 1D NMRspectra processing and quantification. PLoS Comput Biol 14(3): e1006018. https://www.doi.org/10.1371/journal.pcbi.1006018). The spectra deconvolution process yields the signals of all spectra. Due to slight changes in measurement conditions, e.g. pH variations, signal positions may vary slightly across spectra. As a consequence, prior to further analysis signals belonging to the same compound have to be aligned across spectra. This is the purpose of the 'speaq' package.

Usage

```
speaq_align(
  feat = gen_feat_mat(spectrum_data),
  maxShift = 50,
  spectrum_data = generate_lorentz_curves_sim(),
  si_size_real_spectrum = length(spectrum_data[[1]]$y_values),
  verbose = TRUE,
  show = FALSE,
  mfrow = c(2, 1)
)
```

Arguments

`feat`	Output of `gen_feat_mat()`.
`maxShift`	Maximum number of points along the "ppm-axis" which a value can be moved by speaq package e.g. 50. 50 is a suitable starting value for plasma spectra with a digital resolution of 128K. Note that this parameter has to be individually optimized depending on the type of analyzed spectra and the digital resolution. For urine which is more prone to chemical shift variations this value most probably has to be increased.
`spectrum_data`	Output of `generate_lorentz_curves()`.
`si_size_real_spectrum`	Number of real data points in your original spectra.

verbose	Whether to print additional information during the alignment process.
show	Whether to plot the original and aligned spectra.
mfrow	Layout to use for the plot. Passed on to par(). Use mfrow = NULL if the plot layout should not be changed.

Value

A matrix containing the integral values of the spectra after alignment.

There is one row per spectrum and one column per ppm value. The entry at position i, j holds the integral value of the signal from spectrum i that has its center at position j after alignment by speaq. If there is no signal with center j in spectrum i, entry i, j is set to NA. The column names of the matrix are the ppm values of the original spectra.

Example return matrix:

```
    ...  3.59  3.55  3.57  3.56  3.55  3.54  3.53
  .-------------------------------------------> PPM
1 | NA   NA    0.20  NA    NA    NA    0.25  NA
2 | NA   NA    0.15  NA    NA    NA    0.13  NA
3 | NA   NA    NA    0.2   NA    NA    0.18  NA
SpNr
```

Author(s)

Initial version from Wolfram Gronwald. Refactored by Tobias Schmidt in 2024.

Examples

```
sim_subset <- metabodecon_file("bruker/sim_subset")
spectrum_data <- generate_lorentz_curves_sim(sim_subset)
feat <- gen_feat_mat(spectrum_data)
maxShift <- 200
M <- speaq_align(feat, maxShift, spectrum_data, show = TRUE)
str(M)
```

---

tmpdir *Temporary Session Directory*

Description

Returns the path to metabodecon's temporary session directory. This directory equals subdirectory 'metabodecon' of R's temporary session directory base::tempdir() plus additional path normalization.

Usage

```
tmpdir(subdir = NULL, create = FALSE)
```

Arguments

subdir	Optional subdirectory within the temporary session directory.
create	Whether to create the directory if it does not yet exist.

Value

Returns the path to the temporary session directory.

See Also

datadir_temp() datadir_persistent()

Examples

```
tmpdir()
tmpdir("simulate_spectra")
```

---

transp *Make transparent*

Description

Make a color transparent by adding an alpha channel.

Usage

```
transp(col = "violet", alpha = 0.08)
```

Arguments

col	Character string specifying the color to make transparent.
alpha	Numeric value between 0 and 1 specifying the transparency level.

Value

A character string representing the color with an alpha channel.

Examples

```
transp("violet", 0.08)
transp("black", 0.5)
```

tree *Print the Structure of a Directory Tree*

Description

Prints the structure of a directory tree up to a specified maximum level of depth. It lists all files and directories under the specified path, displaying them in a tree-like structure.

Usage

```
tree(path, max.level = 2, level = 0, prefix = "")
```

Arguments

path	The root path from which to start listing the directory structure.
max.level	The maximum depth of directories to list.
level	Internal parameter used for recursion, indicating the current level of depth.
prefix	Internal parameter used for formatting the printed tree structure.

Value

NULL, called for its side effect of printing the directory structure.

Examples

```
metabodecon_dir <- system.file(package = "metabodecon")
tree(metabodecon_dir, max.level = 1)
```

width *Calculate the Width of a Numeric Vector*

Description

Calculates the width of a numeric vector by computing the difference between the maximum and minimum values in the vector.

Usage

```
width(x)
```

Arguments

x	A numeric vector.

Value

The width of the vector, calculated as the difference between its maximum and minimum values.

Examples

```
vec <- c(1, 3, 5, 7, 9)
width(vec)
```

Index