Package 'metabodeconplus'

August 9, 2026

Title Deconvolution, Alignment and Model Fitting of 1d NMR Spectra

Version 0.22.0

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>

Description An integrated framework for deconvolution, alignment and postprocessing of 1-dimensional (1d) nuclear magnetic resonance (NMR) spectra, extended with end-to-end model fitting that turns the resulting matrix of aligned signal integrals into classification models. 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 can be found in Haeckl et al. (2021) <doi:10.3390/metabo11070452>. 'metabodeconplus' is the actively developed successor to the 'metabodecon' package and introduces backwards-incompatible API changes.

License GPL (>= 3)

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

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

biocViews NMR, Deconvolution

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.5.0)

Imports mathjaxr, ranger, readJDX, toscutil (>= 2.8.0), withr

Suggests BiocManager, cachem, covr, devtools, diffobj, digest, doParallel, e1071, glmnet, glue, impute, inline, knitr, lifecycle, MassSpecWavelet, mdrb, microbenchmark, multtest, pkgbuild, pkgload, pROC, R.devices, rcmdcheck, remotes, rlang, rmarkdown, rpart, speaq, styler, testthat (>= 3.0.0), usethis, V8, vdiffr, waldo

LazyData true

LazyDataCompression xz

Config/testthat/edition 3

Config/testthat/parallel true

Config/testthat/start-first read_spectrum, download_example_datasets, cache_example_datasets, align, mcmapply, datadir, get_decon_params, generate_lorentz_curves, smooth_signals2, speaq_align

RdMacros mathjaxr

BuildManual TRUE

Language en-US

Additional_repositories https://spang-lab.r-universe.dev

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-08-09 07:30:08 UTC

Contents

`aaa_Get_Started` *Get URL of Metabodecon "Get Started" Page*

Description

`get_started` and `aaa_Get_Started` both return (and optionally open) the URL of the "Get Started" page of the metabodeconplus documentation. The `aaa_Get_Started` version exists, because functions are listed alphabetically in the reference manual and we want `get_started` to be shown at the top of the list (i.e., it needs to start with an 'a').

Usage

```
aaa_Get_Started(open_browser = interactive())

get_started(open_browser = interactive())
```

Arguments

`open_browser`	If TRUE, the "Get Stated" page is opened in the default browser.

Value

A character string containing the URL of the "Get Started" page.

Author(s)

2024-2025 Tobias Schmidt: initial version.

Examples

```
get_started(open_browser = FALSE)
get_started()
```

`align` *Align deconvoluted spectra*

Description

Aligns peaks across a set of deconvoluted spectra by chaining two stages:

1. **CluPA** (`clupa()`) shifts peak centers continuously toward a reference using hierarchical-clustering FFT segment shifts (Beirnaert et al. 2018, Vu et al. 2011). Adds `x0al` and `pcial` (post-CluPA center and column index) to each peak; original `x0`, `A`, `lambda`, `pcide` are preserved.
2. **Reference snapping** (`snap_to_ref()`) records, for each peak, the nearest reference column within `maxCombine` as `pcisn` / `x0sn`. Peaks farther than `maxCombine` from every reference column get `pcisn = NA` / `x0sn = NA`. No peaks are dropped and amplitudes are not summed here — collisions on the same `pcisn` are aggregated downstream by `si_mat()`.

All spectra in `x` must already live on the same chemical-shift grid (identical `$cs` vector across spectra). Call `harmonize_grid()` upstream if your inputs come from different acquisitions with slight calibration offsets.

Usage

```
align(
  x,
  y = NULL,
  ref = NULL,
  maxShift = 50,
  maxCombine = 0,
  verbose = TRUE,
  nworkers = 1,
  full = TRUE,
  use_speaq = FALSE,
  gap_tol = NULL
)
```

Arguments

`x`	A `decons2` (or `aligns`) object.
`y`	Optional factor of class labels (length `length(x)`). Unused by the default pipeline; accepted for signature compatibility.
`ref`	Optional reference spectrum (`align` or `decon2`). When `NULL` (default) the reference is chosen internally.

maxShift	Maximum number of datapoints a peak center may be shifted by CluPA. maxShift = 0L skips CluPA (sets x0al = x0).
maxCombine	Maximum snap distance for reference snapping in chemical-shift columns. maxCombine = 0L skips snapping. A negative value is treated as maxShift.
verbose	Print progress messages?
nworkers	Number of parallel workers.
full	If TRUE also recompute the aligned superposition during CluPA. Reference snapping always drops sit$supal (the post-snap peak list is no longer Lorentz-compatible).
use_speaq	Use speaq::hClustAlign instead of the bundled CluPA implementation. Defaults to FALSE; the bundled implementation is byte-equivalent to the speaq one (see tests/testthat/test-speaq.R). Setting TRUE requires the suggested speaq package.
gap_tol	Optional gap tolerance in ppm. NULL (default) uses the standard CluPA + snapping pipeline; only consulted by experimental snap backends.

Value

An object of class aligns.

Author(s)

2024-2026 Tobias Schmidt: initial version.

Examples

```
decons <- deconvolute(sim[1:5], sfr=c(3.55, 3.35), verbose=FALSE)
aligned <- align(decons, maxShift=50, maxCombine=20, verbose=FALSE)
```

---

alignment_funs *Alignment building blocks*

Description

Pluggable alignment stages used by align() and the align_fun argument of fit_mdm() / benchmark().

- clupa(): **CluPA** — hierarchical-clustering peak alignment (recursive FFT segment shifts, Beirnaert et al. 2018, Vu et al. 2011). Operates on the Lorentz reconstruction sit$sup already attached to each spectrum by deconvolution.
- snap_to_ref(): **Reference snapping** — snap each peak to the nearest reference column within maxCombine.

All these functions require every input spectrum to share the same $cs grid; an explicit stop() is raised otherwise. Call harmonize_grid() upstream to enforce that invariant.

snap_to_ref() applies the snapping step on its own: for each peak in each spectrum, finds the nearest reference column on the shared cs grid and records that column as pcisn (and its ppm

value as x0sn). Peaks farther than maxCombine columns from every reference column get pcisn = NA / x0sn = NA. Original x0, x0al, A, lambda, pcide and pcial are preserved — snapping only *adds* the snapped fields. Collisions on the same pcisn column are not merged here; si_mat() sums their areas when rasterising the feature matrix. sit$supal is cleared because the post-snap superposition would need recomputing.

Usage

```
clupa(
  x,
  y = NULL,
  ref = NULL,
  maxShift = 50,
  verbose = TRUE,
  nworkers = 1,
  full = TRUE,
  use_speaq = FALSE,
  gap_tol = NULL
)

snap_to_ref(x, ref = NULL, maxCombine = 20, ...)
```

Arguments

x	A decons2 or aligns object.
y	Optional factor of class labels. Unused by the default pipeline; accepted for signature compatibility.
ref	Optional reference spectrum (align or decon2). When NULL, chosen internally.
maxShift	Maximum CluPA shift in datapoints.
verbose	Print progress messages?
nworkers	Number of parallel workers.
full	If TRUE also recompute the aligned superposition.
use_speaq	Use speaq::hClustAlign (CluPA only).
gap_tol	Optional gap tolerance in ppm; only consulted by experimental snap backends.
maxCombine	Maximum reference-snapping distance in datapoints.
...	Ignored.

Value

An object of class aligns.

Examples

```
decons <- deconvolute(sim[1:5], sfr=c(3.55, 3.35), verbose=FALSE)
aligned <- clupa(decons, maxShift=50, verbose=FALSE)  # CluPA stage
snapped <- snap_to_ref(aligned, maxCombine=20)        # reference snapping
```

bin *Bin a spectra-like object into a feature matrix*

Description

Bins the per-spectrum signal vector left-to-right into chunks of `maxCombine` chemical-shift columns and returns the per-bin sums as a feature matrix. Columns whose chemical-shift falls inside any `igrs` interval are removed before binning.

Accepts three input types:

- `spectra`: uses `x[[i]]$si` directly.
- `decons2`: uses `x[[i]]$sit$sup` (smoothed reconstruction).
- `aligns`: builds a sparse vector from `lcpar$pcial` / `lcpar$A` * `pi`, then bins.

Suitable as the `feat_fun` argument of `fit_mdm()` for binning baselines.

Usage

```
bin(x, maxCombine = 128, igrs = list(), peakPos = NULL, ...)
```

Arguments

`x`	A `spectra`, `decons2` or `aligns` object.
`maxCombine`	Bin width in chemical-shift columns.
`igrs`	List of two-element ppm intervals to ignore.
`peakPos`	Optional integer column indices for predict mode; when `NULL` the non-zero bins are kept and attached as `attr(., "peakPos")`.
`...`	Ignored (protocol compatibility with peak_mat).

Value

A numeric matrix with one row per spectrum and one column per bin.

Examples

```
decons <- deconvolute(sim[1:3], sfr=c(3.55, 3.35), verbose=FALSE)
X <- bin(decons, maxCombine=50)
```

bin700 *700-bin Zacharias 2013 feature matrix*

Description

Builds a feature matrix on the fixed 700-bin grid of Zacharias (2013): 300 bins covering 6.5-9.5 ppm + 400 bins covering 0.5-4.5 ppm, both at 0.01 ppm width. The water region (4.5-6.5 ppm) is excluded. Bins are ordered high-to-low — column 1 covers (9.49, 9.50) ppm, column 700 covers (0.50, 0.51) ppm.

Suitable as the `feat_fun` argument of `fit_mdm()`. Per-spectrum dispatch:

- If `lcpar` is empty (raw spectra): bin `$si` directly.
- If `lcpar` is non-empty (deconvoluted / aligned spectra): reconstruct `si_hat = lorentz_sup(cs, lcpar)` on the spectrum's own `cs` grid using `x0al` when present (else `x0`), then bin the reconstruction.

Always returns all 700 columns; `maxCombine`, `igrs` and `peakPos` are accepted for `feat_fun` protocol compatibility but ignored (the bin layout is hardcoded).

Usage

```
bin700(x, maxCombine = 0L, igrs = list(), peakPos = NULL, ...)
```

Arguments

`x`	A `spectra`, `decons2`, or `aligns` object.
`maxCombine`	Ignored. Accepted for protocol compatibility.
`igrs`	Ignored. Accepted for protocol compatibility.
`peakPos`	Ignored. Accepted for protocol compatibility.
`...`	Ignored.

Value

A numeric matrix with one row per spectrum and 700 columns of bin sums. Column names are `sprintf("%.4f", bin_midpoint)`; row names are spectrum names.

Author(s)

2026 Tobias Schmidt: initial version.

Examples

```
# `sim` spans only ~3.3-3.6 ppm, so most of the 700 fixed bins are 0.
X <- bin700(sim[1:3])
```

check_mdrb *Check Rust Backend Availability*

Description

`check_mdrb()` returns a boolean indicating whether a suitable version of the metabodeconplus Rust backend mdrb is currently installed. The Rust backend is entirely optional; metabodeconplus's pure-R backend is the default and always available.

Usage

```
check_mdrb(stop_on_fail = FALSE)
```

Arguments

`stop_on_fail`	If TRUE, an error is thrown if the check fails, providing instructions on how to install mdrb.

Value

`check_mdrb()` returns TRUE if a suitable version of mdrb is installed, else FALSE.

Author(s)

2024-2025 Tobias Schmidt: initial version.

Examples

```
check_mdrb()
```

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.

Author(s)

2024-2025 Tobias Schmidt: initial version.

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 metabodeconplus's data directory*

Description

Returns the path to the directory where `download_example_datasets()` stores metabodeconplus'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.

Author(s)

2024-2025 Tobias Schmidt: initial version.

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 metabodeconplus'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.

Author(s)

2024-2025 Tobias Schmidt: initial version.

See Also

`datadir()`, `datadir_temp()`

Examples

```
datadir_persistent()
```

---

datadir_temp *Temporary Data Directory*

Description

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

Usage

```
datadir_temp()
```

Value

Returns the path to the temporary data directory.

Author(s)

2024-2025 Tobias Schmidt: initial version.

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.

Returns the default grid of (nfit, smit, smws, delta) combinations used by deconvolute() when npmax >= 1. Useful as the deg argument to fit_mdm().

Usage

```
deconvolute(
  x,
  nfit = 3,
  smit = 2,
  smws = 5,
  delta = 6.4,
  npmax = 0,
  sfr = NULL,
  igrs = list(),
  use_rust = FALSE,
  verbose = TRUE,
  nworkers = 1
)

get_deg(conf = "default")
```

Arguments

x	A spectrum or spectra object as described in metabodeconplus-classes.
nfit	Integer. Number of iterations for approximating the parameters for the Lorentz curves. See 'Details'.
smit	Integer. Number of smoothing iterations. See 'Details'.
smws	Integer. Smoothing window size (number of data points; 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'.
npmax	Integer scalar in {-2, -1, 0, 1, 2, ...} controlling how (nfit, smit, smws, delta) are chosen. If npmax >= 1, those four arguments are ignored and a grid search over predefined parameter combinations is performed instead — the combination with the smallest residual area ratio and fewer than npmax peaks is selected. Grid search results are cached to disk automatically. npmax = 0 (default) disables the grid search and uses the literal (nfit, smit, smws, delta) arguments. npmax = -1 is "auto": resolved up front to a single integer (the median

per-spectrum Kneedle elbow on $deg) and broadcast to every spectrum. npmax = -2 is "intrinsic": resolved per spectrum to that spectrum's own Kneedle elbow, so different spectra get different npmax values.

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'.

igrs — Ignore regions. List of length-2 numeric vectors specifying the start and end-points of the chemical shift regions to ignore during deconvolution. Peaks whose centers fall inside any ignore region are excluded from fitting.

use_rust — Controls the deconvolution backend. FALSE or any numeric value < 1 (default) uses the R implementation. TRUE or any numeric value >= 1 uses the Rust backend via mdrb. NULL auto-detects: uses Rust if available, otherwise R. When set to TRUE / >= 1 and mdrb is not installed, an error is thrown.

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.

conf — Character string selecting a configuration. Currently only "default" is supported.

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.

Value

A 'decon2' object as described in metabodeconplus-classes.

A data frame with columns nfit, smit, smws, delta.

Author(s)

2024-2025 Tobias Schmidt: initial version.

Examples

```
## Deconvolute a single spectrum
spectrum <- sim[[1]]
decon <- deconvolute(spectrum)

## Read multiple spectra from disk and deconvolute at once
spectra_dir <- metabodeconplus_file("sim_subset")
spectra <- read_spectra(spectra_dir)
decons <- deconvolute(spectra, sfr = c(3.55,3.35))
get_deg()
```

download_example_datasets

Download metabodeconplus Example Datasets

Description

Downloads example datasets that can be used to test the functionality of the metabodeconplus 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.

Author(s)

2024-2025 Tobias Schmidt: initial version.

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.

[Experimental]

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, 1.1, 1.1),
  sf_vert = "auto",
  si_line = list(),
  sm_line = list(),
  sp_line = list(),
  d2_line = list(),
  al_line = list(),
  lc_lines = list(),
  tp_lines = list(),
  al_lines = list(),
  cent_pts = list(),
  bord_pts = list(),
  norm_pts = list(),
  tp_pts = list(),
  fp_pts = list(),
  miss_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(),
  bt_text = list(),
  lt_text = list(),
  tp_text = list(),
  rt_text = list(),
  tp_verts = list(),
  lc_verts = list(),
  al_verts = list(),
  ze_hline = list(),
  al_arrows = list(),
  lgd = list()
)
```

Arguments

`obj`	An object of type `spectrum` or `decon2`. For details see metabodeconplus-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 `graphics::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.
`sf_vert`	Scale factor for vertical lines corresponding to `lc_verts`, `tp_verts` and `al_verts`. If a numeric value is provided, the height of each line equals the area of the corresponding lorentzian curve multiplied by `sf_vert`. In addition, the following strings are supported: • `"auto"`: A suitable numeric value for `sf_vert` is chosen automatically, in a way that the highest integral equals the highest signal intensity after multiplication with `sf_vert`.

- "peak": Vertical lines are drawn from bottom to top of the corresponding peak.
- "full": Vertical lines are drawn over the full vertical range of the plot region.

si_line, sm_line, sp_line, al_line, d2_line, lc_lines, tp_lines, al_lines
: List of parameters passed to graphics::lines() when drawing the raw signal intensities (si_line), smoothed signal intensities (sm_line), superposition of lorentzian curves (sp_line), aligned lorentzian curves (al_line), second derivative (d2_line), lorentzian curves found by deconvolution (lc_lines), true lorentzian curves (tp_lines) and aligned lorentzian curves (al_lines), respectively.

cent_pts, tp_pts, fp_pts, miss_pts, bord_pts, norm_pts
: List of parameters passed to graphics::points() when drawing the peak center points, true positive peaks, false positive peaks, missed peaks, peak border points and non-peak points.

bg_rect, lc_rects, foc_rect, tp_rects
: List of parameters passed to graphics::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 graphics::axis() when drawing the bottom, left, top and right axis. In addition to the parameters of graphics::axis(), the following additional parameters are supported as well:

  - 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 set the corresponding margin text in a way that reflects the scaling. Example: by default, a scaling factor of 1e6 is used for drawing signal intensities and a scaling factor of 1 for drawing the second derivative. To make clear, that the user should be careful when interpreting the signal intensity values, the corresponding margin text is set to "Signal Intensity [au]" where "au" means "Arbitrary Units", indicating that the values might be scaled.

bt_text, lt_text, tp_text, rt_text
: List of parameters used to overwrite the default values passed to graphics::mtext() when drawing the bottom, left, top and right margin texts (i.e. the axis labels).

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

ze_hline
: List of parameters passed to graphics::abline() when drawing a horizontal line at y = 0.

al_arrows
: List of parameters passed to graphics::arrows() when drawing arrows between the estimated and aligned lorentzian curve centers.

lgd
: List of parameters passed to graphics::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` or `digits` is omitted, a suitable value is chosen automatically. Providing a vector of `digits` causes each digit to be tried 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 representations are tried:

digit	label 1	label 2	label 3	label 4
1	1.0	1.0	1.0	1.0
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.

Author(s)

2024-2025 Tobias Schmidt: initial version.

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, add = FALSE, lgd = FALSE,
              fig = c(.2, .8, .2, .4), mar = c( 0,  0,  0,  0))
draw_spectrum(obj = decon, add = TRUE, lgd = FALSE,
              fig = c(0.2, 0.8, 0.6, 0.8), mar = c(0, 0, 0, 0))
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 `base::setwd()`)
- global options (set via `base::options()`)
- graphical parameters (set via `graphics::par()`)

In addition to that, `evalwith` allows to:

- Redirect or capture the output and/or message stream via `base::sink()`
- Measure the runtime of the evaluated expression via `base::system.time()`
- Creating a temporary test directory (inside `tmpdir()`) and populating it with input files according to `inputs`
- Predefine answers for calls to `base::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 `base::options()`.
pars	Named list of parameters to be set. See `graphics::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 `base::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.

Author(s)

2024-2025 Tobias Schmidt: initial version.

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)
```

---

get_data_dir *Retrieve directory path of an example dataset*

Description

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

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

[Deprecated]

Usage

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

Arguments

`dataset_name`	Either `""`, `"test"`, `"blood"`, `"urine"` or `"aki"`.
`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.

Author(s)

2024-2025 Tobias Schmidt: initial version.

See Also

`download_example_datasets()`

Examples

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

`harmonize_grid` *Harmonize a corpus of spectra onto a shared chemical-shift grid*

Description

Pre-aligns every spectrum in `x` to a single shared chemical-shift grid by integer-datapoint shifting. After the call, every spectrum's `$cs` is bit-identical to the chosen target grid, so downstream code (deconvolution, `clupa()`, `snap_to_ref()`, feature-matrix builders) can index everyone by datapoint and treat `cs` as a single shared variable.

All input spectra must share the same point count and the same ppm width (calibration offsets are allowed, frequency-domain resolution differences are not). For each spectrum the integer offset from the target grid is computed in datapoints, `$si` is rolled by that many positions, the vacated edge is filled with `pad` (default `0`), and `$cs` is replaced by the target grid.

This kills the per-spectrum absolute-calibration offset between acquisitions (typically a constant shift in ppm coming from spectrometer reference setup) so that downstream CluPA only has to correct the remaining sub-datapoint residual + the random chemical-shift drift from sample composition.

Sub-datapoint residual: at most ±0.5 datapoint per spectrum, which is two orders of magnitude smaller than typical Lorentzian peak widths (lambda ~ 1e-3 to 1e-2 ppm vs. spacing ~ 1e-4 ppm), so the rounding error is irrelevant for any downstream fit.

Edge handling: a spectrum that needs to shift right by `k` datapoints loses `k` datapoints from one end and gains `k` `pad` values on the other. For typical NMR spectra the edges are noise, so zero-padding is equivalent to dropping the noise — no real signal is harmed. For large absolute shifts (tens of datapoints) you can verify the discarded region is noise by inspecting `$cs` against the metabolite range of interest.

Usage

```
harmonize_grid(x, target = "median", pad = 0)
```

Arguments

`x`	A `spectra` object (or list of `spectrum` objects).
`target`	Either `"median"` (default) — anchor the target grid at the median first-ppm across `x` — or a numeric vector of length `length(x[[1]]$cs)` giving an explicit target grid.
`pad`	Numeric scalar used to fill the vacated edge after the shift. Default `0`. Use `NA` if you want downstream code to detect the borrowed region explicitly.

Value

A `spectra` object with every spectrum's `$cs` replaced by the target grid and `$si` shifted accordingly. All other fields (`$meta`, `$lcpar`, `$sit`, ...) are preserved unchanged.

Author(s)

2026 Tobias Schmidt: initial version.

Examples

```
# `sim` already shares one cs grid, so harmonization is a no-op here;
# on a corpus from different acquisitions it snaps them to a common grid.
x <- harmonize_grid(sim)
all(sapply(x, function(s) identical(s$cs, x[[1]]$cs)))  # TRUE
```

`headtail` *Show head and tail rows of a matrix-like object*

Description

Returns the first and last `n` rows of a matrix or data frame. If the input has fewer than `2*n` rows, overlapping rows are returned only once.

Usage

```
headtail(x, n = 6)
```

Arguments

`x`	A matrix or data frame.
`n`	Number of rows to take from the top and bottom.

Value

A subset of `x` containing head and tail rows.

Author(s)

2024-2026 Tobias Schmidt: initial version.

Examples

```
x <- matrix(seq_len(30), nrow = 10)
headtail(x, n = 2)
```

heat_spectra *Plot Spectra Heatmap*

Description

Plot a set of spectra as a heatmap. Each row corresponds to one spectrum, each column to a chemical-shift datapoint, and the signal intensity is color-coded.

If the spectra were simulated (i.e. carry a `simpar` element in `meta`), the true peaks are highlighted with thick rectangles spanning `x0 +- lambda`. If the spectra have additionally been deconvoluted, the rectangles are colored according to whether each peak was correctly identified (green), missed (yellow) or wrongly identified (red, drawn at the position of the deconvoluted peak).

Usage

```
heat_spectra(
  objs,
  foc_rgn = NULL,
  what = NULL,
  cols = NULL,
  xlab = "Chemical Shift [ppm]",
  ylab = "Spectrum",
  mar = c(4.1, 2.1, 1.1, 0.5),
  y = NULL,
  y_cols = NULL,
  true_x0 = NULL,
  true_col = "darkgreen",
  true_tol = NULL,
  scale_cols = FALSE,
  cex_names = 0.8,
  xaxis_side = 1,
  col_scores = NULL,
  col_sep = NULL,
  row_sep = NULL,
  main = NULL,
  names = NULL,
  ref = NULL,
  ref_col = "red",
  ref_lwd = NULL,
  sparse = FALSE,
  xaxt = "s"
)
```

Arguments

`objs`	An object of type `spectrum`, `spectra`, `decon2`, `decons2`, `align` or `aligns`, OR a numeric matrix with chemical-shift values as `colnames` (rows are spectra). To plot a feature matrix derived from peak areas, use `si_mat()` and pass the result.

foc_rgn	Numeric vector of length 2 specifying the focus region in ppm (e.g. c(3.55, 3.35)). If NULL (default), the full spectrum is shown.
what	Which signal to plot: "si" (raw), "sup" (superposition of Lorentz curves) or "supal" (aligned superposition). Defaults to a sensible choice based on the input class. Ignored when objs is a matrix.
cols	Character vector of colors used as intensity color palette. Defaults to hcl.colors(64, "YlOrRd", rev = TRUE).
xlab, ylab	Axis labels.
mar	Numeric vector of length 4 specifying the plot margins. Passed to graphics::par(). The right margin is overridden at runtime to fit the spectra names.
y	Optional vector of class labels (one per spectrum). If provided, the spectra names are colored according to the class.
y_cols	Character vector of colors used to color the spectra names by class. Defaults to rainbow(nlevels(as.factor(y))). Ignored if y is NULL.
true_x0	Optional numeric vector of true peak positions (in ppm). If supplied, the x-axis tick labels of columns within true_tol ppm of any true_x0 are drawn in true_col. Useful to highlight known discriminating features in a sparse feature matrix from si_mat().
true_col	Color used for x-axis labels of columns close to a true_x0 value.
true_tol	Numeric tolerance in ppm for matching columns to true_x0. Defaults to half the median column spacing.
scale_cols	If TRUE, scale each column (chemical shift) to a symmetric range before mapping to colours, so per-feature contrasts are comparable.
cex_names	Character expansion factor for the spectrum name labels drawn on the right side of the heatmap. Defaults to 0.8.
xaxis_side	On which side to draw the x-axis: 1 (bottom, default) or 3 (top).
col_scores	Optional numeric vector of length ncol(Z) (after foc_rgn filtering) giving a per-column score (e.g. lasso coefficients or feature importances). When supplied, columns are sorted by col_scores (ascending) and the score is appended in brackets to each x-axis label.
col_sep	Vertical column separators. NULL (default) draws a separator at the sign change of col_scores if given, otherwise none. FALSE disables separators entirely. An integer vector draws separators *after* the given (post-sort) column indices.
row_sep	Horizontal row separators. NULL (default) draws separators at class changes when y is given, otherwise none. FALSE disables separators entirely. An integer vector draws separators *after* the given row indices.
main	Optional plot title. Drawn via graphics::title().
names	Controls per-spectrum names drawn on the right side. NULL (default) or TRUE uses the spectrum names. FALSE hides them and shrinks the right margin accordingly. A character vector overrides the labels.
ref	Integer row index (1-based) of a reference spectrum to highlight with a rectangle, or NULL (default) for no highlight.
ref_col	Color of the reference-row rectangle. Defaults to "red".

ref_lwd	Line width of the reference-row rectangle. Defaults to par("lwd").
sparse	If TRUE, render the heatmap as a sparse peak matrix: all cells are zero except at the columns where a peak center sits (picked from lcpar$pcisn / lcpar$pcial / lcpar$pcide, in that priority). The value at a peak column is the Lorentzian peak height A / lambda; collisions on the same column are summed. Ignored when objs is a matrix.
xaxt	Character. "s" (default) draws the x-axis normally; "n" suppresses x-axis ticks, tick labels and xlab.

Value

NULL. Called for side effect of plotting.

Author(s)

2024-2026 Tobias Schmidt: initial version.

Examples

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

---

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 metabodeconplus-classes.

Author(s)

2024-2025 Tobias Schmidt: initial version.

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 = TRUE)
```

mdm *Metabodecon Models*

Description

[Experimental]

Fit (`fit_mdm()`) or cross-validate (`benchmark()`) a binary classification model built on a set of NMR spectra. Both run the full deconvolute -> align -> snap -> featurize -> fit pipeline with sensible defaults and expose only the parameters a typical user tunes; the classification backend is chosen via `model`. Power users who need to swap individual pipeline stages can call the internal engines `fit_mdm_internal()` / `benchmark_internal()`, which take pluggable `decon_fun` / `align_fun` / `snap_fun` / `feat_fun` / `fit_fun` / `predict_fun` arguments.

`fit_mdm()` runs the pipeline once, or iterates over the cartesian product of `npmax` / `maxShift` / `maxCombine` when any is a vector and returns the row with the highest `acc` (ties broken by `auc`). `benchmark()` wraps `fit_mdm()` in outer k-fold cross-validation to estimate end-to-end performance on held-out spectra.

Usage

```
fit_mdm(
  x,
  y,
  model = c("ranger", "lasso"),
  npmax = -1L,
  maxShift = -1L,
  maxCombine = 10L,
  nworkers = 1L,
  seed = 1L,
  verbosity = 1L,
  ...
)

benchmark(
  x,
  y,
  model = c("ranger", "lasso"),
  npmax = -1L,
  maxShift = -1L,
  maxCombine = 10L,
  nworkers = 1L,
  seed = 1L,
  verbosity = 2L,
  k = 3L,
  ...
)
```

Arguments

x	Spectra object.
y	Factor vector with class labels for each spectrum.
model	Classification backend. One of "ranger" (default, probability random forest) or "lasso" (L1-penalised logistic regression via glmnet).
npmax	Max peaks per spectrum. Integer in {-2, -1, 0, 1, ...}, scalar or vector. -1 (default) selects the median per-spectrum Kneedle elbow; -2 selects each spectrum's own elbow.
maxShift	Max CluPA shift in datapoints. Integer >= -1, scalar or vector. -1 (default) means auto (sweep to the alignment-correlation dip).
maxCombine	Reference-snapping window in datapoints. Integer, scalar or vector. Default 10.
nworkers	Number of workers for deconvolution, alignment and the inner fitter.
seed	Random seed. Forwarded to the fitter; also used for stratified fold assignment inside benchmark(). May be a vector for repeated CV.
verbosity	Verbosity level.
...	Further arguments passed on to the internal engine (fit_mdm_internal() / benchmark_internal()), e.g. sfr, igrs, deg, use_rust. Rarely needed.
k	Number of outer folds for benchmark().

Value

fit_mdm() returns an object of class mdm with elements model (trained backend model of the best grid row), ref (a list list(align, snap) for prediction-time replay), params (resolved pipeline parameters of the best row), the scalar performance of the best row (acc, auc, acc_se, auc_se), and mog (the augmented grid with per-row performance).

benchmark() returns a list with elements models (one fitted model per outer fold), predictions (per-spectrum out-of-fold predictions), performance (per-fold acc / auc) and overall (pooled acc / auc).

Examples

```
# Small, fast illustrative run. `deg` restricts deconvolution to a single
# parameter set and scalar npmax/maxShift/maxCombine give a one-row model
# grid (`mog`); benchmark() does a single 2-fold cross-validation round.
i <- c(1:3, 51:53)                      # 3 spectra per class
x <- sim2[i]
y <- attr(sim2, "group")[i]
deg <- expand.grid(nfit=3, smit=1, smws=3, delta=1.6)
m <- fit_mdm(x, y, npmax=10L, maxShift=1L, maxCombine=2L, deg=deg, verbosity=0)
bm <- benchmark(x, y, npmax=10L, maxShift=1L, maxCombine=2L, deg=deg, k=2L, verbosity=0)
# `model = "lasso"` selects L1-penalised logistic regression instead.
```

mdm_methods *S3 methods for mdm objects*

Description

[Experimental]

WARNING: These methods are experimental and must not be used in production. Their API is very likely to change in non-backwards-compatible ways over the next few weeks.

S3 methods for objects of class `mdm` and `summary.mdm`.

`predict.mdm()` predicts probabilities, classes, link scores, or all three from an `mdm` object. When `newdata` is a spectra object, the spectra are deconvoluted, aligned and snapped to the references stored in the model before prediction. When `newdata` is a numeric matrix, it is used directly as the feature matrix.

`print.mdm()` prints a compact model summary.

`coef.mdm()` returns lasso coefficients (or ranger importance).

`plot.mdm()` plots the lasso path (or ranger importance bars).

`summary.mdm()` builds a compact summary list.

`print.summary.mdm()` prints formatted output for `summary.mdm` objects.

Usage

```
## S3 method for class 'mdm'
predict(
  object,
  newdata,
  type = c("all", "prob", "class", "link"),
  nworkers = 1,
  verbosity = 1,
  ...
)

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

## S3 method for class 'mdm'
coef(object, ...)

## S3 method for class 'mdm'
plot(x, ...)

## S3 method for class 'mdm'
summary(object, ...)

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

Arguments

object, x	A fitted mdm object (for predict, coef, summary, print and plot) or a summary.mdm object (for print.summary.mdm).
newdata	Spectra object or numeric feature matrix.
type	Prediction type, one of "all", "prob", "class", "link".
nworkers	Number of workers to deconvolute and align newdata.
verbosity	Integer verbosity level.
...	Passed to underlying methods where applicable.

Value

- predict: numeric vector of probabilities, classes, and/or link scores.
- print: invisibly returns x.
- coef: coefficient object from glmnet (or ranger importance).
- plot: invisibly returns NULL.
- summary: object of class summary.mdm.
- print.summary.mdm: invisibly returns x.

---

metabodeconplus-classes

Metabodecon Classes and Helpers

Description

Metabodecon represents NMR data using a small set of S3 classes connected by **cumulative inheritance**. A raw spectrum has class "spectrum". After deconvolute() it gains class "decon2", so its class vector becomes c("decon2", "spectrum"). After align() it gains class "align", with class vector c("align", "decon2", "spectrum"). The corresponding collection classes follow the same pattern.

Every deconvoluted or aligned object is still a spectrum in the base::inherits() sense, so S3 generic behavior for spectrum or spectra also works at every stage. Element order may vary between versions; always access fields by name, e.g. x$si or x[["cs"]]. Elements marked optional may be absent or NULL.

Usage

```
is_spectrum(x)
is_spectra(x)
as_spectra(x, ...)
as_decon2(x)
as_decons2(x)
get_names(x, default = "spectrum_\045d")
```

Arguments

x	A metabodeconplus object, collection, list of objects, or path.
default	Used by get_names() when no object names are present.
...	Parameters passed to read_spectrum() when x is a path.

Value

is_spectrum() and is_spectra() return TRUE or FALSE. The as_*() functions return an object of the requested class. get_names() returns a character vector.

Singlet classes

- spectrum: A single NMR spectrum. Class vector: "spectrum". Constructed by read_spectrum(), make_spectrum(), or simulate_spectrum(). Carries the fields under *Always present (spectrum)* below.
- decon2: A single deconvoluted NMR spectrum. Class vector: c("decon2", "spectrum"). Produced by deconvolute(). In addition to the spectrum fields, a decon2 carries the *Added by deconvolute()* fields below.
- align: A single deconvoluted NMR spectrum whose peak positions have been aligned across a collection. Class vector: c("align", "decon2", "spectrum"). Produced by align(). Carries everything a decon2 does, plus the *Added by align()* fields below.

Collection classes

For each singlet class there is a collection class that wraps a list of those singlets:

- spectra: List of spectrum. Class vector "spectra".
- decons2: List of decon2. Class vector c("decons2", "spectra").
- aligns: List of align. Class vector c("aligns", "decons2", "spectra").

Collections inherit from "spectra", so generic methods written for spectra also work on decons2 and aligns. Constructed by read_spectra() (returns spectra), deconvolute() when given a spectra (returns decons2), and align() (returns aligns). Concatenation follows the cumulative rule: the result class is the most-general, least-specific class among the inputs. Mixing an align with a plain decon2 yields decons2; mixing any plain spectrum in yields spectra.

Always present (spectrum)

1. cs: Vector of chemical shifts in ppm. Same length as si.
2. si: Vector of signal intensities (au). si[i] is the intensity at cs[i].
3. meta: Optional list of metadata, e.g. name (spectrum name), path (source path), type (experiment type), fq (signal frequencies in Hz), mfs (magnetic field strength), or simpar (true Lorentz-curve parameters for simulated spectra).

Added by deconvolute()

A decon2 object additionally has:

1. args: List of deconvolution parameters used (nfit, smit, smws, delta, sfr, igrs, npmax, use_rust, verbose).
2. sit: Data frame of signal intensities after transformations: sm (smoothed), sup (superposition of fitted Lorentz curves), and supal (superposition of aligned Lorentz curves, added by align()).
3. peak: Data frame of peak triplets with columns center, left, right: integer indices into cs.
4. lcpar: Data frame of Lorentz-curve parameters. Always carries x0 (center in ppm), A (amplitude), lambda (half-width) and pcide (integer column index into cs for x0). After clupa() also x0al / pcial (post-CluPA center and cs index). After snap_to_ref() also x0sn / pcisn (post-snap center and cs index, with NA for peaks snapped beyond maxCombine). A and lambda are preserved through every stage.

Added by align()

An align object has the same fields as decon2, but with the alignment slots populated:

- lcpar$x0al: Peak Centers after CluPA alignment in ppm
- lcpar$pcial: Peak Centers after CluPA alignment as cs indices
- lcpar$x0sn: Peak Centers after reference snapping in ppm (NA when snapped out)
- lcpar$pcisn: Peak Centers after reference snapping as cs indices (NA when snapped out)
- sit$supal: Signal Intensities of the superposition of aligned Lorentz curves

Predicates

is_spectrum() and is_spectra() test inheritance from the base metabodeconplus classes. Since decon2 and align inherit from spectrum, and decons2 and aligns inherit from spectra, they satisfy these checks. To test for a specific lifecycle stage, use base::inherits() directly, e.g. inherits(x, "decon2") or inherits(x, "aligns").

Converters

as_spectra() turns a path, spectrum, or list of spectrum objects into a spectra collection. as_decon2() and as_decons2() are identity converters that validate their input.

Naming helpers

get_names() returns collection names by checking each element's metadata, each element's direct name, the list names, and finally generated default names.

Author(s)

2024-2025 Tobias Schmidt: initial version.

Examples

```
s <- sim[[1]]
inherits(s, "spectrum")
is_spectrum(s)

d <- deconvolute(s, sfr = c(3.55, 3.35))
class(d) # c("decon2", "spectrum")
inherits(d, "spectrum") # TRUE

ds <- deconvolute(sim[1:3], sfr = c(3.55, 3.35))
class(ds)              # c("decons2", "spectra")
as_spectra(s)
get_names(list(s, myspec = s))
```

`metabodeconplus_file` *Return Path to File or Directory in metabodeconplus Package*

Description

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

Usage

```
metabodeconplus_file(name = "sim_01")
```

Arguments

`name`	The name to search for.

Value

The file or directory path.

Author(s)

2024-2025 Tobias Schmidt: initial version.

Examples

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

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

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

---

peak_mat *Peak feature matrix*

Description

Thin wrapper around `si_mat()` suitable as the `feat_fun` argument of `fit_mdm()`. Equivalent to `si_mat(x, igrs=igrs)`; the snapping that used to live here has moved into `align()` (reference-snapping stage).

Usage

```
peak_mat(x, igrs = list(), peakPos = NULL, ...)
```

Arguments

`x`	An `aligns` object (or `decons2`).
`igrs`	List of two-element ppm intervals to ignore.
`peakPos`	Optional integer column indices, forwarded to `si_mat()` for predict mode.
`...`	Ignored. Accepted so `peak_mat` and `bin()` share a single `feat_fun(x, maxCombine, igrs)` protocol; `peak_mat` ignores `maxCombine` because snapping happens upstream inside `align()`.

Value

A numeric matrix with spectra in rows and chemical shifts as colnames.

Author(s)

2024-2026 Tobias Schmidt: initial version.

Examples

```
decons <- deconvolute(sim[1:3], sfr=c(3.55, 3.35), verbose=FALSE)
aligned <- align(decons, maxShift=50, maxCombine=20, verbose=FALSE)
X <- peak_mat(aligned)
```

plot_spectra *Plot Spectra*

Description

Plot a set of deconvoluted spectra.

Usage

```
plot_spectra(
  x,
  foc_rgn = NULL,
  what = NULL,
  sfy = 1e+06,
  cols = NULL,
  lty = NULL,
  names = NULL,
  xlab = "Chemical Shift [ppm]",
  ylab = paste("Signal Intensity [au] /", sfy),
  mar = c(4.1, 4.1, 1.1, 0.1),
  lgd = TRUE,
  main = NULL,
  xaxt = "s",
  yaxt = "s"
)
```

Arguments

x	An object of type decons0, decons1 or decons2. For details see metabodeconplus-classes.
foc_rgn	Numeric vector of length 2 specifying the focus region in ppm (e.g. c(3.55, 3.35)). If NULL (default), the full spectrum is shown.
what	Which signal to plot: "supal" (aligned superposition, default with fallback to "sup" then "si"), "sup" (superposition) or "si" (raw).
sfy	Scaling factor for the y-axis.
cols	Character vector of colors, one per spectrum. Defaults to rainbow(n).
lty	Line type(s), one per spectrum. Recycled if shorter than n. Defaults to 1 (solid) for all spectra.
names	Character vector of legend labels. Defaults to spectrum names.
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.
lgd	Logical or list. If TRUE, a legend is drawn at "topright" with cex = 0.8. If a list, its elements are passed to graphics::legend() to override position, size, etc. Pass lgd = FALSE to hide.

main	Optional plot title. Drawn via graphics::title().
xaxt, yaxt	Character. "s" (default) draws the axis normally; "n" suppresses axis ticks and tick labels. Passed to graphics::plot().

Value

A plot of the deconvoluted spectra.

Author(s)

2024-2025 Tobias Schmidt: initial version.

See Also

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

Examples

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

---

plot_spectrum *Plot Spectrum*

Description

Plot a spectrum and zoom in on a specific region.

[Experimental]

Usage

```
plot_spectrum(
  x,
  ...,
  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, decon2 or align. For details see metabodeconplus-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 specifying the margins of the plot. Passed to graphics::par(). If set to NULL, a suitable value is chosen automatically.
frame	A list of values passed to graphics::box() when drawing the frame around plot region. If set to NULL, no frame is drawn.
con_lines	A list of values passed to graphics::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.

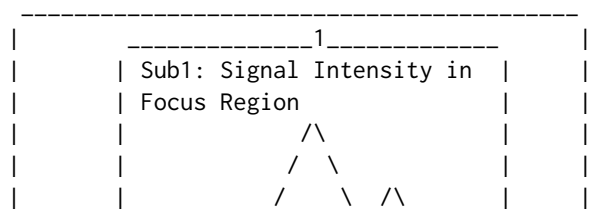
Details

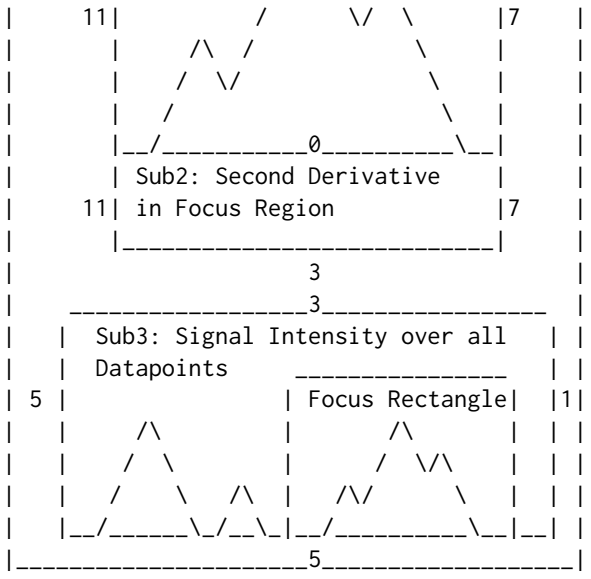
This function first initializes a new plotting canvas. After that it calls draw_spectrum() multiple times to draw the following sub figures onto the plotting canvas:

1. The signal intensities in the focus region
2. The second derivative in the focus region
3. The signal intensities over all datapoints

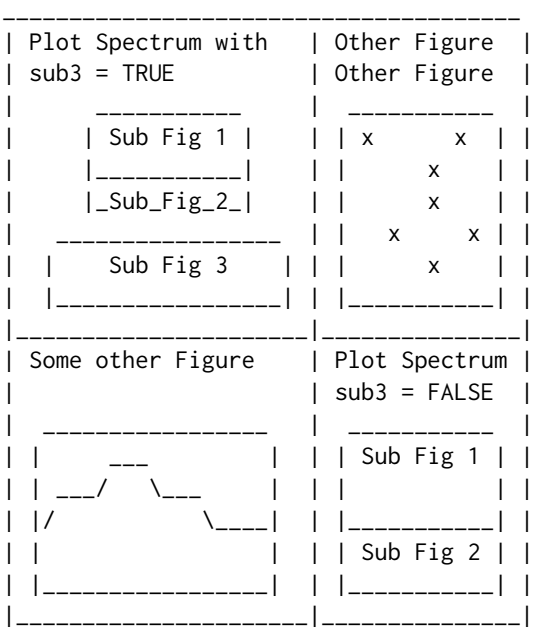
The argument lists for the individual calls to draw_spectrum() are determined at runtime and depend on the arguments passed to plot_spectrum() as well as the currently active graphics device. To customize the appearance of the individual sub plots, you can overwrite each value passed to draw_spectrum() 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 `graphics::par()`. Example:



Value

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

Author(s)

2024-2025 Tobias Schmidt: initial version.

Examples

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

spec <- sim[[1]]
decon <- deconvolute(sim[1], sfr = c(3.55, 3.35))

## 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_text = 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)))
```

`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 Get Started article.
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 metabodeconplus-classes.

Author(s)

2024-2025 Tobias Schmidt: initial version.

Examples

```
relpath <- "example_datasets/bruker/urine"
urine <- system.file(relpath, package = "metabodeconplus")
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 = metabodeconplus_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 Get Started article.
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 metabodeconplus-classes.

Author(s)

2024-2025 Tobias Schmidt: initial version.

Examples

```
relpath <- "example_datasets/bruker/urine"
urine <- system.file(relpath, package = "metabodeconplus")
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 metabodeconplus-classes.

sim2 *The Sim2 Classification Dataset*

Description

A simulated two-group classification dataset for demonstrating `fit_mdm()` and `benchmark()`. It contains 100 simulated 1D NMR spectra split evenly into groups `A` and `B`, where 3 out of 25 peaks per spectrum differ between the groups: in group `A`, two peaks are scaled by `1.25` and one peak by `1/1.25` (≈ 0.80). Group `B` is left unmodified. The first spectrum (`sim2_001`) is constructed without any global or per-peak ppm jitter so it can serve as a clean unshifted alignment reference.

Usage

```
sim2
```

Format

A `spectra` object consisting of 100 `spectrum` objects, where each spectrum contains 2048 datapoints ranging from 3.59 to 3.28 ppm. The per-spectrum group labels are attached as `attr(sim2, "group")`, a named factor with levels `A` and `B`. For details about `spectrum` and `spectra` objects see metabodeconplus-classes.

Each spectrum's `meta$simpar` carries the standard fields (`x0`, `A`, `lambda`, `noise`) plus five sim2-specific fields: `base_x0` (the 25 reference peak positions, identical across spectra), `dx0` (per-peak jitter in ppm), `gx0` (scalar global ppm shift), `diff_AB` (integer indices into `base_x0` of the peaks that differ between groups), and `ab_factors` (the multiplicative factors applied to those peaks in group A). They satisfy `x0[k] = base_x0[k] + dx0[k] + gx0`.

`attr(sim2, "true_x0")` is a numeric vector with the post-alignment ppm positions of the discriminating peaks (one per `diff_AB` index), useful for highlighting them on plots of aligned feature matrices.

Details

Peak parameters (positions, areas, half-widths and noise) were chosen to match the values recovered by deconvoluting the sim dataset, which itself is derived from the Blood reference dataset (see sim). Concretely:

- 25 base peaks per spectrum with positions drawn uniformly in `[3.37, 3.52]` ppm. The reference-grid step is `0.00015 ppm/dp`.
- Per-peak jitter with standard deviation 4 datapoints (≈ 0.00060 ppm) plus a per-spectrum global ppm shift with standard deviation 8 datapoints (≈ 0.00120 ppm) to mimic chemical shift variation between samples.
- Base areas drawn from a log-normal distribution centered around `2500` (in ppm-area units) and varied per spectrum by `+/-60%`.
- Base half-widths drawn uniformly in `[0.0009, 0.0013]` ppm and varied per spectrum by `+/-10%`.
- Gaussian noise with standard deviation `2200`.

- In group A, three base peaks (indices simpar$diff_AB) have their areas multiplied by c(1.25, 1.25, 1/1.25) (stored in simpar$ab_factors). Group B is left unmodified. Spectrum sim2_001 is generated with dx0 = 0 and gx0 = 0 to provide a clean unshifted alignment reference.

---

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 metabodeconplus-classes.

Author(s)

2024-2025 Tobias Schmidt: initial version.

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()
```

---

si_mat *Signal-integral matrix*

Description

Builds a per-spectrum peak-area matrix. Each row is a spectrum, each column is a chemical-shift datapoint. For each peak, the column is picked from lcpar$pcisn (post-snap) when available, else lcpar$pcial (post-CluPA), else lcpar$pcide (post-decon). Peaks with pcisn = NA (snapped beyond maxCombine) are skipped. Collisions on the same column have their A * pi summed.

si_mat() is intentionally a dumb peak-list rasterizer: all alignment (continuous shift via CluPA) and reference snapping must have happened upstream — typically inside align(). To build a feature matrix where every spectrum shares the same column grid, run align(x, maxShift, maxCombine) first.

Usage

```
si_mat(x, drop_zero = FALSE, igrs = list(), peakPos = NULL, ...)
```

Arguments

x	A decons2 or aligns object.
drop_zero	Drop columns whose entries are all zero?
igrs	List of two-element ppm intervals to zero out before returning.
peakPos	Optional integer column indices. When supplied (predict mode) the matrix is subset to those columns; when NULL and used as a feat_fun the non-zero columns are kept and attached as attr(., "peakPos").
...	Ignored (protocol compatibility with other feat_funs).

Value

A numeric matrix with one row per spectrum and `length(x[[1]]$cs)` columns (the full cs grid). Column names are ppm values; row names are spectrum names.

Author(s)

2024-2026 Tobias Schmidt: initial version.

Examples

```
decons <- deconvolute(sim[1:2], sfr=c(3.55, 3.35), verbose=FALSE)
aligned <- align(decons, maxShift=50, maxCombine=20, verbose=FALSE)
X <- si_mat(aligned)
```

---

tmpdir *Temporary Session Directory*

Description

Returns the path to metabodeconplus's temporary session directory. This directory equals subdirectory 'metabodeconplus' 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.

Author(s)

2024-2025 Tobias Schmidt: initial version.

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.

Author(s)

2024-2025 Tobias Schmidt: initial version.

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,
  max.entries = Inf,
  show.counts = FALSE,
  files.first = FALSE,
  level = 0,
```

```
  prefix = ""
)

tree_preview(
  path,
  max.level = 1,
  max.entries = 9,
  show.counts = TRUE,
  files.first = TRUE
)
```

Arguments

`path`	The root path from which to start listing the directory structure.
`max.level`	The maximum depth of directories to list.
`max.entries`	Maximum number of children to print per directory. If a directory has more entries than this limit, the first `ceiling(max.entries / 2)` and last `floor(max.entries / 2)` children are shown, with `... [N]` in between, where `N` is the number of omitted children.
`show.counts`	Logical. If `TRUE`, prints the number of direct children (files + subdirectories) in brackets after each directory name. Disabled by default.
`files.first`	Logical. If `TRUE`, files are listed before subdirectories. If `FALSE` (default), subdirectories are listed first.
`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.

Author(s)

2024-2025 Tobias Schmidt: initial version.

Examples

```
metabodeconplus_dir <- system.file(package = "metabodeconplus")
tree(metabodeconplus_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.

Author(s)

2024-2025 Tobias Schmidt: initial version.

Examples

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

Index