Package 'spant'

July 10, 2020

Type Package

Title MR Spectroscopy Analysis Tools

Version 1.7.0

Date 2020-07-09

Description Tools for reading, visualising and processing Magnetic Resonance Spectroscopy data. <https://martin3141.github.io/spant/>.

BugReports https://github.com/martin3141/spant/issues

License GPL-3

RoxygenNote 7.1.1

NeedsCompilation yes

LazyData yes

Depends R (>= 2.10)

Imports abind, plyr, foreach, pracma, stringr, complexplus, signal, matrixcalc, minpack.lm, nnls, utils, graphics, grDevices, smoother, readr, magrittr, ptw, viridisLite, mmand, RNifti, RNiftyReg, fields, MASS, shiny, miniUI, oro.dicom, numDeriv, nloptr, irlba, tibble, jsonlite

Suggests neurobase, oro.nifti, knitr, rmarkdown, testthat, doParallel

VignetteBuilder knitr

Encoding UTF-8

Language en-GB

Author Martin Wilson [cre, aut],
Yong Wang [ctb],
John Muschelli [ctb]

Maintainer Martin Wilson <martin@pipegrep.co.uk>

Repository CRAN

Date/Publication 2020-07-09 23:00:09 UTC

R topics documented:

spant-package *spant: spectroscopy analysis tools.*

Description

spant provides a set of tools for reading, visualising and processing Magnetic Resonance Spectroscopy (MRS) data.

Details

To get started with spant, take a look at the introduction vignette:

```
vignette("spant-intro",package="spant")
```

Full list of vignettes:

```
browseVignettes(package = "spant")
```

Full list of functions:

```
help(package = spant,help_type = "html")
```

An online version of the documentation is available from:

https://martin3141.github.io/spant/

Author(s)

Maintainer: Martin Wilson <martin@pipegrep.co.uk>

Other contributors:

- Yong Wang [contributor]
- John Muschelli [contributor]

See Also

Useful links:

- Report bugs at https://github.com/martin3141/spant/issues

abfit_opts *Return a list of options for an ABfit analysis.*

Description

Return a list of options for an ABfit analysis.

Usage

```
abfit_opts(
  init_damping = 5,
  maxiters = 1024,
  max_shift = 10,
  max_damping = 15,
  max_phase = 360,
  lambda = NULL,
  ppm_left = 4,
  ppm_right = 0.2,
  zp = TRUE,
  bl_ed_pppm = 2,
  auto_bl_flex = TRUE,
  bl_comps_pppm = 15,
  export_sp_fit = FALSE,
  max_asym = 0.25,
  max_basis_shift = 1,
  max_basis_damping = 2,
  maxiters_pre = 1000,
  algo_pre = "NLOPT_LN_NELDERMEAD",
  min_bl_ed_pppm = NULL,
  max_bl_ed_pppm = 7,
  auto_bl_flex_n = 20,
  pre_fit_bl_ed_pppm = 1,
  remove_lip_mm_prefit = FALSE,
  pre_align = TRUE,
  max_pre_align_shift = 0.1,
  pre_align_ref_freqs = c(2.01, 3.03, 3.22),
  noise_region = c(-0.5, -2.5),
  optimal_smooth_criterion = "maic",
  aic_smoothing_factor = 5,
  anal_jac = TRUE,
  pre_fit_ppm_left = 4,
  pre_fit_ppm_right = 1.8,
  phi1_optim = FALSE,
  phi1_init = 0,
  max_dphi1 = 0.2,
  max_basis_shift_broad = 1,
  max_basis_damping_broad = 2,
```

```
  ahat_calc_method = "lh_pnnls"
)
```

Arguments

`init_damping`	initial value of the Gaussian global damping parameter (Hz). Very poorly shimmed or high field data may benefit from a larger value.
`maxiters`	The maximum number of iterations to run for the detailed fit.
`max_shift`	The maximum allowable shift to be applied in the optimisation phase of fitting (Hz).
`max_damping`	maximum permitted value of the global damping parameter (Hz).
`max_phase`	maximum permitted value of the global zero-order phase term (degrees).
`lambda`	manually set the the baseline smoothness parameter.
`ppm_left`	downfield frequency limit for the fitting range (ppm).
`ppm_right`	upfield frequency limit for the fitting range (ppm).
`zp`	zero pad the data to twice the original length before fitting.
`bl_ed_pppm`	manually set the the baseline smoothness parameter (ED per ppm).
`auto_bl_flex`	automatically determine the level of baseline smoothness.
`bl_comps_pppm`	spline basis density (signals per ppm).
`export_sp_fit`	add the fitted spline functions to the fit result.
`max_asym`	maximum allowable value of the asymmetry parameter.
`max_basis_shift`	maximum allowable frequency shift for individual basis signals (Hz).
`max_basis_damping`	maximum allowable Lorentzian damping factor for individual basis signals (Hz).
`maxiters_pre`	maximum iterations for the coarse (pre-)fit.
`algo_pre`	optimisation method for the coarse (pre-)fit.
`min_bl_ed_pppm`	minimum value for the candidate baseline flexibility analyses (ED per ppm).
`max_bl_ed_pppm`	minimum value for the candidate baseline flexibility analyses (ED per ppm).
`auto_bl_flex_n`	number of candidate baseline analyses to perform.
`pre_fit_bl_ed_pppm`	level of baseline flexibility to use in the coarse fitting stage of the algorithm (ED per ppm).
`remove_lip_mm_prefit`	remove broad signals in the coarse fitting stage of the algorithm.
`pre_align`	perform a pre-alignment step before coarse fitting.
`max_pre_align_shift`	maximum allowable shift in the pre-alignment step (ppm).
`pre_align_ref_freqs`	a vector of prominent spectral frequencies used in the pre-alignment step (ppm).
`noise_region`	spectral region to estimate the noise level (ppm).

`optimal_smooth_criterion`
: method to determine the optimal smoothness.

`aic_smoothing_factor`
: modification factor for the AIC calculation.

`anal_jac`
: use a analytical approximation to the jacobian in the detailed fitting stage.

`pre_fit_ppm_left`
: downfield frequency limit for the fitting range in the coarse fitting stage of the algorithm (ppm).

`pre_fit_ppm_right`
: upfield frequency limit for the fitting range in the coarse fitting stage of the algorithm (ppm).

`phi1_optim`
: apply and optimise a frequency dependant phase term.

`phi1_init`
: initial value for the frequency dependant phase term (ms).

`max_dphi1`
: maximum allowable change from the initial frequency dependant phase term (ms).

`max_basis_shift_broad`
: maximum allowable shift for broad signals in the basis (Hz). Determined based on their name beginning with Lip or MM.

`max_basis_damping_broad`
: maximum allowable Lorentzian damping for broad signals in the basis (Hz). Determined based on their name beginning with Lip or MM.

`ahat_calc_method`
: method to calculate the metabolite amplitudes. May be one of: "lh_pnnls" or "ls".

Value

full list of options.

Examples

```
opts <- abfit_opts(ppm_left = 4.2, noise_region = c(-1, -3))
```

---

acquire *Simulate pulse sequence acquisition.*

Description

Simulate pulse sequence acquisition.

Usage

```
acquire(sys, rec_phase = 180, tol = 1e-04, detect = NULL)
```

Arguments

sys	spin system object.
rec_phase	receiver phase in degrees.
tol	ignore resonance amplitudes below this threshold.
detect	detection nuclei.

Value

a list of resonance amplitudes and frequencies.

---

align *Align spectra to a reference frequency using a convolution based method.*

Description

Align spectra to a reference frequency using a convolution based method.

Usage

```
align(
  mrs_data,
  ref_freq = 4.65,
  zf_factor = 2,
  lb = 2,
  max_shift = 20,
  ret_df = FALSE
)
```

Arguments

mrs_data	data to be aligned.
ref_freq	reference frequency in ppm units. More than one frequency may be specified.
zf_factor	zero filling factor to increase alignment resolution.
lb	line broadening to apply to the reference signal.
max_shift	maximum allowable shift in Hz.
ret_df	return frequency shifts in addition to aligned data (logical).

Value

aligned data object.

apodise_xy *Apodise MRSI data in the x-y direction with a k-space hamming filter.*

Description

Apodise MRSI data in the x-y direction with a k-space hamming filter.

Usage

```
apodise_xy(mrs_data)
```

Arguments

mrs_data	MRSI data.

Value

apodised data.

append_basis *Combine a pair of basis set objects.*

Description

Combine a pair of basis set objects.

Usage

```
append_basis(basis_a, basis_b)
```

Arguments

basis_a	first basis.
basis_b	second basis.

Value

combined basis set object.

append_coils *Append MRS data across the coil dimension, assumes they matched across the other dimensions.*

Description

Append MRS data across the coil dimension, assumes they matched across the other dimensions.

Usage

```
append_coils(...)
```

Arguments

...	MRS data objects as arguments, or a list of MRS data objects.

Value

a single MRS data object with the input objects concatenated together.

append_dyns *Append MRS data across the dynamic dimension, assumes they matched across the other dimensions.*

Description

Append MRS data across the dynamic dimension, assumes they matched across the other dimensions.

Usage

```
append_dyns(...)
```

Arguments

...	MRS data objects as arguments, or a list of MRS data objects.

Value

a single MRS data object with the input objects concatenated together.

apply_axes *Apply a function over specified array axes.*

Description

Apply a function over specified array axes.

Usage

```
apply_axes(x, axes, fun, ...)
```

Arguments

x	an array.
axes	a vector of axes to apply fun over.
fun	function to be applied.
...	optional arguments to fun.

Value

array.

Examples

```
z <- array(1:1000, dim = c(10, 10, 10))
a <- apply_axes(z, 3, fft)
a[1,1,] == fft(z[1,1,])
a <- apply_axes(z, 3, sum)
a[1,1,] == sum(z[1,1,])
```

apply_mrs *Apply a function across given dimensions of a MRS data object.*

Description

Apply a function across given dimensions of a MRS data object.

Usage

```
apply_mrs(mrs_data, dims, fun, ..., data_only = FALSE)
```

Arguments

mrs_data	MRS data.
dims	dimensions to apply the function.
fun	name of the function.
...	arguments to the function.
data_only	return an array rather than an MRS data object.

---

apply_pvc *Convert default LCM/TARQUIN concentration scaling to molal units with partial volume correction.*

Description

Convert default LCM/TARQUIN concentration scaling to molal units with partial volume correction.

Usage

```
apply_pvc(fit_result, p_vols, te, tr)
```

Arguments

fit_result	a fit_result object to apply partial volume correction.
p_vols	a numeric vector of partial volumes.
te	the MRS TE.
tr	the MRS TR.

Value

a fit_result object with a rescaled results table.

Arg.mrs_data *Apply Arg operator to an MRS dataset.*

Description

Apply Arg operator to an MRS dataset.

Usage

```
## S3 method for class 'mrs_data'
Arg(z)
```

Arguments

z	MRS data.

Value

MRS data following Arg operator.

`array2mrs_data` *Convert a 7 dimensional array in into a mrs_data object. The array dimensions should be ordered as : dummy, X, Y, Z, dynamic, coil, FID.*

Description

Convert a 7 dimensional array in into a mrs_data object. The array dimensions should be ordered as : dummy, X, Y, Z, dynamic, coil, FID.

Usage

```
array2mrs_data(
  data_array,
  fs = def_fs(),
  ft = def_ft(),
  ref = def_ref(),
  fd = FALSE
)
```

Arguments

data_array	7d data array.
fs	sampling frequency in Hz.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.
fd	flag to indicate if the matrix is in the frequency domain (logical).

Value

mrs_data object.

`auto_phase` *Perform zeroth-order phase correction based on the minimisation of the squared difference between the real and magnitude components of the spectrum.*

Description

Perform zeroth-order phase correction based on the minimisation of the squared difference between the real and magnitude components of the spectrum.

Usage

```
auto_phase(mrs_data, xlim = NULL, ret_phase = FALSE)
```

Arguments

`mrs_data`	an object of class `mrs_data`.
`xlim`	frequency range (default units of PPM) to including in the phase.
`ret_phase`	return phase values (logical).

Value

MRS data object and phase values (optional).

`back_extrap` *Back extrapolate time-domain points using the Burg autoregressive model*

Description

Back extrapolate time-domain points using the Burg autoregressive model

Usage

```
back_extrap(mrs_data, n_pts)
```

Arguments

`mrs_data`	mrs_data object.
`n_pts`	number of points to extrapolate.

Value

back extrapolated data.

`basis2mrs_data` *Convert a basis object to an mrs_data object - where basis signals are spread across the dynamic dimension.*

Description

Convert a basis object to an mrs_data object - where basis signals are spread across the dynamic dimension.

Usage

```
basis2mrs_data(basis, sum_elements = FALSE, amp = NULL, shift = NULL)
```

Arguments

`basis`	basis set object.
`sum_elements`	return the sum of basis elements (logical)
`amp`	a vector of scaling factors to apply to each basis element.
`shift`	a vector of frequency shifts (in PPM) to apply to each basis element.

Value

an mrs_data object with basis signals spread across the dynamic dimension or summed.

`bbase` *Generate a spline basis, slightly adapted from : "Splines, knots, and penalties", Eilers 2010.*

Description

Generate a spline basis, slightly adapted from : "Splines, knots, and penalties", Eilers 2010.

Usage

```
bbase(N, number, deg = 3)
```

Arguments

`N`	number of data points.
`number`	number of spline functions.
`deg`	spline degree : deg = 1 linear, deg = 2 quadratic, deg = 3 cubic.

Value

spline basis as a matrix.

bc_als *Baseline correction using the ALS method.*

Description

Baseline correction using the ALS method.

Usage

```
bc_als(mrs_data, lambda = 10000, p = 0.001)
```

Arguments

mrs_data	mrs_data object.
lambda	lambda parameter.
p	p parameter.

Value

baseline corrected data.

bc_constant *Remove a constant baseline offset based on a reference spectral region.*

Description

Remove a constant baseline offset based on a reference spectral region.

Usage

```
bc_constant(mrs_data, xlim)
```

Arguments

mrs_data	MRS data.
xlim	spectral range containing a flat baseline region to measure the offset.

Value

baseline corrected data.

`beta2lw` *Covert a beta value in the time-domain to an equivalent linewidth in Hz: x * exp(-i * t * t * beta).*

Description

Covert a beta value in the time-domain to an equivalent linewidth in Hz: x * exp(-i * t * t * beta).

Usage

```
beta2lw(beta)
```

Arguments

`beta`	beta damping value.

Value

linewidth value in Hz.

`calc_coil_noise_cor` *Calculate the noise correlation between coil elements.*

Description

Calculate the noise correlation between coil elements.

Usage

```
calc_coil_noise_cor(noise_data)
```

Arguments

`noise_data`	`mrs_data` object with one FID for each coil element.

Value

correlation matrix.

calc_coil_noise_sd *Calculate the noise standard deviation for each coil element.*

Description

Calculate the noise standard deviation for each coil element.

Usage

```
calc_coil_noise_sd(noise_data)
```

Arguments

noise_data	mrs_data object with one FID for each coil element.

Value

array of standard deviations.

calc_ed_from_lambda *Calculate the effective dimensions of a spline smoother from lambda.*

Description

Calculate the effective dimensions of a spline smoother from lambda.

Usage

```
calc_ed_from_lambda(spline_basis, deriv_mat, lambda)
```

Arguments

spline_basis	spline basis.
deriv_mat	derivative matrix.
lambda	smoothing parameter.

Value

the effective dimension value.

`calc_peak_info_vec` *Calculate the FWHM of a peak from a vector of intensity values.*

Description

Calculate the FWHM of a peak from a vector of intensity values.

Usage

```
calc_peak_info_vec(data_pts, interp_f)
```

Arguments

`data_pts`	input vector.
`interp_f`	interpolation factor to improve the FWHM estimate.

Value

a vector of: x position of the highest data point, maximum peak value in the y axis, FWHM in the units of data points.

`calc_sd_poly` *Perform a polynomial fit, subtract and return the standard deviation of the residuals.*

Description

Perform a polynomial fit, subtract and return the standard deviation of the residuals.

Usage

```
calc_sd_poly(y, degree = 1)
```

Arguments

`y`	array.
`degree`	polynomial degree.

Value

standard deviation of the fit residuals.

`calc_spec_diff` *Calculate the sum of squares differences between two mrs_data objects.*

Description

Calculate the sum of squares differences between two mrs_data objects.

Usage

```
calc_spec_diff(mrs_data, ref = NULL, xlim = c(4, 0.5))
```

Arguments

`mrs_data`	mrs_data object.
`ref`	reference mrs_data object to calculate differences.
`xlim`	spectral limits to perform calculation.

Value

an array of the sum of squared difference values.

`calc_spec_snr` *Calculate the spectral SNR.*

Description

SNR is defined as the maximum signal value divided by the standard deviation of the noise.

Usage

```
calc_spec_snr(
  mrs_data,
  sig_region = c(4, 0.5),
  noise_region = c(-0.5, -2.5),
  p_order = 2,
  interp_f = 4,
  full_output = FALSE
)
```

Arguments

mrs_data	an object of class mrs_data.
sig_region	a ppm region to define where the maximum signal value should be estimated.
noise_region	a ppm region to defined where the noise level should be estimated.
p_order	polynomial order to fit to the noise region before estimating the standard deviation.
interp_f	interpolation factor to improve detection of the highest signal value.
full_output	output signal, noise and SNR values separately.

Details

The mean noise value is subtracted from the maximum signal value to reduce DC offset bias. A polynomial detrending fit (second order by default) is applied to the noise region before the noise standard deviation is estimated.

Value

an array of SNR values.

---

check_lcm *Check LCModel can be run*

Description

Check LCModel can be run

Usage

```
check_lcm()
```

---

check_tqn *Check the TARQUIN binary can be run*

Description

Check the TARQUIN binary can be run

Usage

```
check_tqn()
```

collapse_to_dyns *Collapse MRS data by concatenating spectra along the dynamic dimension.*

Description

Collapse MRS data by concatenating spectra along the dynamic dimension.

Usage

```
collapse_to_dyns(x)

## S3 method for class 'mrs_data'
collapse_to_dyns(x)

## S3 method for class 'fit_result'
collapse_to_dyns(x)
```

Arguments

x	data object to be collapsed (mrs_data or fit_result object).

Value

collapsed data with spectra or fits concatenated along the dynamic dimension.

comb_coils *Combine coil data based on the first data point of a reference signal.*

Description

By default, elements are phased and scaled prior to summation. Where a reference signal is not given, the mean dynamic signal will be used instead.

Usage

```
comb_coils(metab, ref = NULL, noise = NULL, scale = TRUE, sum_coils = TRUE)
```

Arguments

metab	MRS data containing metabolite data.
ref	MRS data containing reference data (optional).
noise	MRS data from a noise scan (optional).
scale	option to rescale coil elements based on the first data point (logical).
sum_coils	sum the coil elements as a final step (logical).

Value

MRS data.

`comb_coils_fp_pc` *Combine coil data following phase correction based on the first data point in the FID.*

Description

Combine coil data following phase correction based on the first data point in the FID.

Usage

```
comb_coils_fp_pc(metab, ref = NULL, sum_coils = TRUE, ret_ref = FALSE)
```

Arguments

`metab`	MRS data containing metabolite data.
`ref`	MRS data containing reference data (optional).
`sum_coils`	sum the coil elements as a final step (logical).
`ret_ref`	return the reference data following correction.

Value

MRS data.

`comb_csv_results` *Combine the results from multiple csv format files into a table.*

Description

Combine the results from multiple csv format files into a table.

Usage

```
comb_csv_results(pattern, supp_mess = TRUE, ...)
```

Arguments

`pattern`	glob string to match csv files.
`supp_mess`	suppress messages from the read_csv function.
`...`	extra parameters to pass to read_csv.

Value

results table.

comb_fits — *Combine a list of fit_result objects into a single fit object containing a dynamic series.*

Description

Combine a list of fit_result objects into a single fit object containing a dynamic series.

Usage

```
comb_fits(fit_list)
```

Arguments

fit_list	list of fit_result objects.

Value

fit_result object.

comb_metab_ref — *Combine a reference and metabolite mrs_data object.*

Description

Combine a reference and metabolite mrs_data object.

Usage

```
comb_metab_ref(metab, ref)
```

Arguments

metab	metabolite mrs_data object.
ref	reference mrs_data object.

Value

combined metabolite and reference mrs_data object.

Conj.mrs_data *Apply Conj operator to an MRS dataset.*

Description

Apply Conj operator to an MRS dataset.

Usage

```
## S3 method for class 'mrs_data'
Conj(z)
```

Arguments

z	MRS data.

Value

MRS data following Conj operator.

conv_mrs *Convolve two MRS data objects.*

Description

Convolve two MRS data objects.

Usage

```
conv_mrs(mrs_data, conv)
```

Arguments

mrs_data	MRS data to be convolved.
conv	convolution data stored as an mrs_data object.

Value

convolved data.

crop_spec *Crop* mrs_data *object based on a frequency range.*

Description

Crop mrs_data object based on a frequency range.

Usage

```
crop_spec(mrs_data, xlim = c(4, 0.2), scale = "ppm")
```

Arguments

mrs_data	MRS data.
xlim	range of values to crop in the spectral dimension eg xlim = c(4, 0.2).
scale	the units to use for the frequency scale, can be one of: "ppm", "hz" or "points".

Value

cropped mrs_data object.

crop_td_pts *Crop* mrs_data *object data points in the time-domain.*

Description

Crop mrs_data object data points in the time-domain.

Usage

```
crop_td_pts(mrs_data, start = NULL, end = NULL)
```

Arguments

mrs_data	MRS data.
start	starting data point (defaults to 1).
end	ending data point (defaults to the last saved point).

Value

cropped mrs_data object.

crop_xy — *Crop an MRSI dataset in the x-y direction*

Description

Crop an MRSI dataset in the x-y direction

Usage

```
crop_xy(mrs_data, x_dim, y_dim)
```

Arguments

mrs_data	MRS data object.
x_dim	x dimension output length.
y_dim	y dimension output length.

Value

selected subset of MRS data.

crossprod_3d — *Compute the vector cross product between vectors x and y. Adapted from http://stackoverflow.com/questions/15162741/what-is-rs-crossproduct-function*

Description

Compute the vector cross product between vectors x and y. Adapted from http://stackoverflow.com/questions/15162741/what-is-rs-crossproduct-function

Usage

```
crossprod_3d(x, y)
```

Arguments

x	vector of length 3.
y	vector of length 3.

Value

vector cross product of x and y.

decimate_mrs *Decimate an MRS signal by a factor.*

Description

Decimate an MRS signal by a factor.

Usage

```
decimate_mrs(mrs_data, q = 2)
```

Arguments

mrs_data	MRS data object.
q	integer factor to downsample by (default = 2).

Value

decimated data.

def_acq_paras *Return (and optionally modify using the input arguments) a list of the default acquisition parameters.*

Description

Return (and optionally modify using the input arguments) a list of the default acquisition parameters.

Usage

```
def_acq_paras(
  ft = getOption("spant.def_ft"),
  fs = getOption("spant.def_fs"),
  N = getOption("spant.def_N"),
  ref = getOption("spant.def_ref")
)
```

Arguments

ft	specify the transmitter frequency in Hz.
fs	specify the sampling frequency in Hz.
N	specify the number of data points in the spectral dimension.
ref	specify the reference value for ppm scale.

Value

A list containing the following elements:

- ft transmitter frequency in Hz.
- fs sampling frequency in Hz.
- N number of data points in the spectral dimension.
- ref reference value for ppm scale.

---

def_fs *Return the default sampling frequency in Hz.*

Description

Return the default sampling frequency in Hz.

Usage

```
def_fs()
```

Value

sampling frequency in Hz.

---

def_ft *Return the default transmitter frequency in Hz.*

Description

Return the default transmitter frequency in Hz.

Usage

```
def_ft()
```

Value

transmitter frequency in Hz.

def_N — *Return the default number of data points in the spectral dimension.*

Description

Return the default number of data points in the spectral dimension.

Usage

```
def_N()
```

Value

number of data points in the spectral dimension.

def_nuc — *Return the default nucleus.*

Description

Return the default nucleus.

Usage

```
def_nuc()
```

Value

number of data points in the spectral dimension.

def_ref — *Return the default reference value for ppm scale.*

Description

Return the default reference value for ppm scale.

Usage

```
def_ref()
```

Value

reference value for ppm scale.

`diff_mrs` *Apply the diff operator to an MRS dataset in the FID/spectral dimension.*

Description

Apply the diff operator to an MRS dataset in the FID/spectral dimension.

Usage

```
diff_mrs(mrs_data, ...)
```

Arguments

`mrs_data`	MRS data.
`...`	additional arguments to the diff function.

Value

MRS data following diff operator.

`downsample_mrs` *Downsample an MRS signal by a factor of 2 using an FFT "brick-wall" filter.*

Description

Downsample an MRS signal by a factor of 2 using an FFT "brick-wall" filter.

Usage

```
downsample_mrs(mrs_data)
```

Arguments

`mrs_data`	MRS data object.

Value

downsampled data.

ecc — *Eddy current correction.*

Description

Apply eddy current correction using the Klose method.

Usage

```
ecc(metab, ref, rev = FALSE)
```

Arguments

metab	MRS data to be corrected.
ref	reference dataset.
rev	reverse the correction.

Details

In vivo proton spectroscopy in presence of eddy currents. Klose U. Magn Reson Med. 1990 Apr;14(1):26-30.

Value

corrected data in the time domain.

est_noise_sd — *Estimate the standard deviation of the noise from a segment of an mrs_data object.*

Description

Estimate the standard deviation of the noise from a segment of an mrs_data object.

Usage

```
est_noise_sd(mrs_data, n = 100, offset = 100, p_order = 2)
```

Arguments

mrs_data	MRS data object.
n	number of data points (taken from the end of array) to use in the estimation.
offset	number of final points to exclude from the calculation.
p_order	polynomial order to fit to the data before estimating the standard deviation.

Value

standard deviation array.

`fd2td` *Transform frequency-domain data to the time-domain.*

Description

Transform frequency-domain data to the time-domain.

Usage

```
fd2td(mrs_data)
```

Arguments

`mrs_data`	MRS data in frequency-domain representation.

Value

MRS data in time-domain representation.

`fd_conv_filt` *Frequency-domain convolution based filter.*

Description

Frequency-domain convolution based filter.

Usage

```
fd_conv_filt(mrs_data, K = 25, ext = 1)
```

Arguments

`mrs_data`	MRS data to be filtered.
`K`	window width in data points.
`ext`	point separation for linear extrapolation.

fit_amps — *Extract the fit amplitudes from an object of class* `fit_result`.

Description

Extract the fit amplitudes from an object of class `fit_result`.

Usage

```
fit_amps(
  x,
  inc_index = FALSE,
  sort_names = FALSE,
  append_common_1h_comb = TRUE
)
```

Arguments

`x`	`fit_result` object.
`inc_index`	include columns for the voxel index.
`sort_names`	sort the basis set names alphabetically.
`append_common_1h_comb`	append commonly used 1H metabolite combinations eg tNAA = NAA + NAAG.

Value

a dataframe of amplitudes.

fit_diags — *Calculate diagnostic information for object of class* `fit_result`.

Description

Calculate diagnostic information for object of class `fit_result`.

Usage

```
fit_diags(x, amps = NULL)
```

Arguments

`x`	`fit_result` object.
`amps`	known metabolite amplitudes.

Value

a dataframe of diagnostic information.

`fit_mrs` *Perform a fit based analysis of MRS data.*

Description

Note that TARQUIN and LCModel require these packages to be installed, and the functions set_tqn_cmd and set_lcm_cmd (respectively) need to be used to specify the location of these software packages.

Usage

```
fit_mrs(
  metab,
  basis = NULL,
  method = "ABFIT",
  w_ref = NULL,
  opts = NULL,
  parallel = FALSE,
  time = TRUE,
  progress = "text"
)
```

Arguments

`metab`	metabolite data.
`basis`	basis class object or character vector to basis file in LCModel .basis format.
`method`	'ABFIT' (default), 'VARPRO', 'VARPRO_3P', 'TARQUIN' or 'LCMODEL'.
`w_ref`	water reference data for concentration scaling (optional).
`opts`	options to pass to the analysis method.
`parallel`	perform analyses in parallel (TRUE or FALSE).
`time`	measure the time taken for the analysis to complete (TRUE or FALSE).
`progress`	option is passed to plyr::alply function to display a progress bar during fitting. Default value is "text", set to "none" to disable.

Details

Fitting approaches described in the following references: VARPRO van der Veen JW, de Beer R, Luyten PR, van Ormondt D. Accurate quantification of in vivo 31P NMR signals using the variable projection method and prior knowledge. Magn Reson Med 1988;6:92-98.

TARQUIN Wilson, M., Reynolds, G., Kauppinen, R. A., Arvanitis, T. N. & Peet, A. C. A constrained least-squares approach to the automated quantitation of in vivo 1H magnetic resonance spectroscopy data. Magn Reson Med 2011;65:1-12.

LCModel Provencher SW. Estimation of metabolite concentrations from localized in vivo proton NMR spectra. Magn Reson Med 1993;30:672-679.

Value

MRS analysis object.

Examples

```
fname <- system.file("extdata","philips_spar_sdat_WS.SDAT",package="spant")
svs <- read_mrs(fname, format="spar_sdat")
## Not run:
basis <- sim_basis_1h_brain_press(svs)
fit_result <- fit_mrs(svs, basis)

## End(Not run)
```

---

fit_res2csv *Write fit results table to a csv file.*

Description

Write fit results table to a csv file.

Usage

```
fit_res2csv(fit_res, fname, unscaled = FALSE)
```

Arguments

fit_res	fit result object.
fname	filename of csv file.
unscaled	output the unscaled result table (default = FALSE).

---

fp_phase *Return the phase of the first data point in the time-domain.*

Description

Return the phase of the first data point in the time-domain.

Usage

```
fp_phase(mrs_data)
```

Arguments

mrs_data	MRS data.

Value

phase values in degrees.

`fp_phase_correct` *Perform a zeroth order phase correction based on the phase of the first data point in the time-domain.*

Description

Perform a zeroth order phase correction based on the phase of the first data point in the time-domain.

Usage

```
fp_phase_correct(mrs_data, ret_phase = FALSE)
```

Arguments

`mrs_data`	MRS data to be corrected.
`ret_phase`	return phase values (logical).

Value

corrected data or a list with corrected data and optional phase values.

`fs` *Return the sampling frequency in Hz of an MRS dataset.*

Description

Return the sampling frequency in Hz of an MRS dataset.

Usage

```
fs(mrs_data)
```

Arguments

`mrs_data`	MRS data.

Value

sampling frequency in Hz.

ft_shift — *Perform a fft and ffshift on a vector.*

Description

Perform a fft and ffshift on a vector.

Usage

```
ft_shift(vec_in)
```

Arguments

vec_in	vector input.

Value

output vector.

ft_shift_mat — *Perform a fft and fftshift on a matrix with each column replaced by its shifted fft.*

Description

Perform a fft and fftshift on a matrix with each column replaced by its shifted fft.

Usage

```
ft_shift_mat(mat_in)
```

Arguments

mat_in	matrix input.

Value

output matrix.

gen_F — *Generate the F product operator.*

Description

Generate the F product operator.

Usage

```
gen_F(sys, op, detect = NULL)
```

Arguments

sys	spin system object.
op	operator, one of "x", "y", "z", "p", "m".
detect	detection nuclei.

Value

F product operator matrix.

gen_F_xy — *Generate the Fxy product operator with a specified phase.*

Description

Generate the Fxy product operator with a specified phase.

Usage

```
gen_F_xy(sys, phase, detect = NULL)
```

Arguments

sys	spin system object.
phase	phase angle in degrees.
detect	detection nuclei.

Value

product operator matrix.

get_1h_brain_basis_paras

Return a list of `mol_parameter` *objects suitable for 1H brain MRS analyses.*

Description

Return a list of `mol_parameter` objects suitable for 1H brain MRS analyses.

Usage

```
get_1h_brain_basis_paras(ft, metab_lw = NULL, lcm_compat = FALSE)
```

Arguments

`ft`	transmitter frequency in Hz.
`metab_lw`	linewidth of metabolite signals (Hz).
`lcm_compat`	when TRUE, lipid, MM and -CrCH molecules will be excluded from the output.

Value

list of `mol_parameter` objects.

get_1h_brain_basis_paras_v1

Return a list of `mol_parameter` *objects suitable for 1H brain MRS analyses.*

Description

Return a list of `mol_parameter` objects suitable for 1H brain MRS analyses.

Usage

```
get_1h_brain_basis_paras_v1(ft, metab_lw = NULL, lcm_compat = FALSE)
```

Arguments

`ft`	transmitter frequency in Hz.
`metab_lw`	linewidth of metabolite signals (Hz).
`lcm_compat`	when TRUE, lipid, MM and -CrCH molecules will be excluded from the output.

Value

list of `mol_parameter` objects.

`get_1h_brain_basis_paras_v2`

Return a list of `mol_parameter` *objects suitable for 1H brain MRS analyses.*

Description

Return a list of `mol_parameter` objects suitable for 1H brain MRS analyses.

Usage

```
get_1h_brain_basis_paras_v2(ft, metab_lw = NULL, lcm_compat = FALSE)
```

Arguments

`ft`	transmitter frequency in Hz.
`metab_lw`	linewidth of metabolite signals (Hz).
`lcm_compat`	when TRUE, lipid, MM and -CrCH molecules will be excluded from the output.

Value

list of `mol_parameter` objects.

`get_1h_brain_basis_paras_v3`

Return a list of `mol_parameter` *objects suitable for 1H brain MRS analyses.*

Description

Return a list of `mol_parameter` objects suitable for 1H brain MRS analyses.

Usage

```
get_1h_brain_basis_paras_v3(ft, metab_lw = NULL, lcm_compat = FALSE)
```

Arguments

`ft`	transmitter frequency in Hz.
`metab_lw`	linewidth of metabolite signals (Hz).
`lcm_compat`	when TRUE, lipid, MM and -CrCH molecules will be excluded from the output.

Value

list of `mol_parameter` objects.

get_2d_psf — *Get the point spread function (PSF) for a 2D phase encoded MRSI scan.*

Description

Get the point spread function (PSF) for a 2D phase encoded MRSI scan.

Usage

```
get_2d_psf(FOV = 160, mat_size = 16, sampling = "circ", hamming = FALSE)
```

Arguments

FOV	field of view in mm.
mat_size	acquisition matrix size (not interpolated).
sampling	can be either "circ" for circular or "rect" for rectangular.
hamming	should Hamming k-space weighting be applied (default FALSE).

Value

A matrix of the PSF with 1mm resolution.

get_acq_paras — *Return acquisition parameters from a MRS data object.*

Description

Return acquisition parameters from a MRS data object.

Usage

```
get_acq_paras(mrs_data)
```

Arguments

mrs_data	MRS data.

Value

list of acquisition parameters.

get_dyns *Extract a subset of dynamic scans.*

Description

Extract a subset of dynamic scans.

Usage

```
get_dyns(mrs_data, subset)
```

Arguments

mrs_data	dynamic MRS data.
subset	vector containing indices to the dynamic scans to be returned.

Value

MRS data containing the subset of requested dynamics.

get_even_dyns *Return even numbered dynamic scans starting from 1 (2,4,6...).*

Description

Return even numbered dynamic scans starting from 1 (2,4,6...).

Usage

```
get_even_dyns(mrs_data)
```

Arguments

mrs_data	dynamic MRS data.

Value

dynamic MRS data containing even numbered scans.

get_fh_dyns *Return the first half of a dynamic series.*

Description

Return the first half of a dynamic series.

Usage

```
get_fh_dyns(mrs_data)
```

Arguments

mrs_data	dynamic MRS data.

Value

first half of the dynamic series.

get_fit_map *Get a data array from a fit result.*

Description

Get a data array from a fit result.

Usage

```
get_fit_map(fit_res, name)
```

Arguments

fit_res	fit_result object.
name	name of the quantity to plot, eg "tNAA".

get_fit_table *Combine all fitting data points into a single dataframe.*

Description

Combine all fitting data points into a single dataframe.

Usage

```
get_fit_table(fit_res)
```

Arguments

fit_res	a single fit_result object.

Value

a dataframe containing the fit data points.

get_fp *Return the first time-domain data point.*

Description

Return the first time-domain data point.

Usage

```
get_fp(mrs_data)
```

Arguments

mrs_data	MRS data.

Value

first time-domain data point.

get_guassian_pulse *Generate a gaussian pulse shape.*

Description

Generate a gaussian pulse shape.

Usage

```
get_guassian_pulse(angle, n, trunc = 1)
```

Arguments

angle	pulse angle in degrees.
n	number of points to generate.
trunc	percentage truncation factor.

get_metab *Extract the metabolite component from an mrs_data object.*

Description

Extract the metabolite component from an mrs_data object.

Usage

```
get_metab(mrs_data)
```

Arguments

mrs_data	MRS data.

Value

metabolite component.

get_mol_names — *Return a character array of names that may be used with the* get_mol_paras *function.*

Description

Return a character array of names that may be used with the get_mol_paras function.

Usage

```
get_mol_names()
```

Value

a character array of names.

get_mol_paras — *Get a* mol_parameters *object for a named molecule.*

Description

Get a mol_parameters object for a named molecule.

Usage

```
get_mol_paras(name, ...)
```

Arguments

name	the name of the molecule.
...	arguments to pass to molecule definition function.

get_mrsi2d_seg *Calculate the partial volume estimates for each voxel in a 2D MRSI dataset.*

Description

Localisation is assumed to be perfect in the z direction and determined by the ker input in the x-y direction.

Usage

```
get_mrsi2d_seg(mrs_data, mri_seg, ker)
```

Arguments

mrs_data	2D MRSI data with multiple voxels in the x-y dimension.
mri_seg	MRI data with values corresponding to the segmentation class. Must be 1mm isotropic resolution.
ker	MRSI PSF kernel in the x-y direction compatible with the mmand package, eg: mmand::shapeKernel(c(10, 10), type = "box").

Value

a data frame of partial volume estimates.

get_mrsi_voi *Generate a MRSI VOI from an* mrs_data *object.*

Description

Generate a MRSI VOI from an mrs_data object.

Usage

```
get_mrsi_voi(mrs_data, target_mri = NULL, map = NULL, ker = mmand::boxKernel())
```

Arguments

mrs_data	MRS data.
target_mri	optional image data to match the intended volume space.
map	optional voi intensity map.
ker	kernel to rescale the map data to the target_mri.

Value

volume data as a nifti object.

get_mrsi_voxel *Generate a MRSI voxel from an* mrs_data *object.*

Description

Generate a MRSI voxel from an mrs_data object.

Usage

```
get_mrsi_voxel(mrs_data, target_mri, x_pos, y_pos, z_pos)
```

Arguments

mrs_data	MRS data.
target_mri	optional image data to match the intended volume space.
x_pos	x voxel coordinate.
y_pos	y voxel coordinate.
z_pos	z voxel coordinate.

Value

volume data as a nifti object.

get_mrsi_voxel_xy_psf *Generate a MRSI voxel PSF from an* mrs_data *object.*

Description

Generate a MRSI voxel PSF from an mrs_data object.

Usage

```
get_mrsi_voxel_xy_psf(mrs_data, target_mri, x_pos, y_pos, z_pos)
```

Arguments

mrs_data	MRS data.
target_mri	optional image data to match the intended volume space.
x_pos	x voxel coordinate.
y_pos	y voxel coordinate.
z_pos	z voxel coordinate.

Value

volume data as a nifti object.

get_odd_dyns *Return odd numbered dynamic scans starting from 1 (1,3,5...).*

Description

Return odd numbered dynamic scans starting from 1 (1,3,5...).

Usage

```
get_odd_dyns(mrs_data)
```

Arguments

mrs_data	dynamic MRS data.

Value

dynamic MRS data containing odd numbered scans.

get_ref *Extract the reference component from an mrs_data object.*

Description

Extract the reference component from an mrs_data object.

Usage

```
get_ref(mrs_data)
```

Arguments

mrs_data	MRS data.

Value

reference component.

`get_seg_ind` *Get the indices of data points lying between two values (end > x > start).*

Description

Get the indices of data points lying between two values (end > x > start).

Usage

```
get_seg_ind(scale, start, end)
```

Arguments

`scale`	full list of values.
`start`	smallest value in the subset.
`end`	largest value in the subset.

Value

set of indices.

`get_sh_dyns` *Return the second half of a dynamic series.*

Description

Return the second half of a dynamic series.

Usage

```
get_sh_dyns(mrs_data)
```

Arguments

`mrs_data`	dynamic MRS data.

Value

second half of the dynamic series.

get_slice *Return a single slice from a larger MRSI dataset.*

Description

Return a single slice from a larger MRSI dataset.

Usage

```
get_slice(mrs_data, z_pos)
```

Arguments

mrs_data	MRSI data.
z_pos	the z index to extract.

Value

MRS data.

get_subset *Extract a subset of MRS data.*

Description

Extract a subset of MRS data.

Usage

```
get_subset(
  mrs_data,
  x_set = NULL,
  y_set = NULL,
  z_set = NULL,
  dyn_set = NULL,
  coil_set = NULL
)
```

Arguments

mrs_data	MRS data object.
x_set	x indices to include in the output (default all).
y_set	y indices to include in the output (default all).
z_set	z indices to include in the output (default all).
dyn_set	dynamic indices to include in the output (default all).
coil_set	coil indices to include in the output (default all).

Value

selected subset of MRS data.

`get_svs_voi` *Generate a SVS acquisition volume from an* `mrs_data` *object.*

Description

Generate a SVS acquisition volume from an `mrs_data` object.

Usage

```
get_svs_voi(mrs_data, target_mri)
```

Arguments

`mrs_data`	MRS data.
`target_mri`	optional image data to match the intended volume space.

Value

volume data as a nifti object.

`get_td_amp` *Return an array of amplitudes derived from fitting the initial points in the time domain and extrapolating back to t=0.*

Description

Return an array of amplitudes derived from fitting the initial points in the time domain and extrapolating back to t=0.

Usage

```
get_td_amp(mrs_data, nstart = 10, nend = 50)
```

Arguments

`mrs_data`	MRS data.
`nstart`	first data point to fit.
`nend`	last data point to fit.

Value

array of amplitudes.

get_uncoupled_mol — *Generate a* mol_parameters *object for a simple spin system with one resonance.*

Description

Generate a mol_parameters object for a simple spin system with one resonance.

Usage

```
get_uncoupled_mol(name, chem_shift, nucleus, scale_factor, lw, lg)
```

Arguments

name	name of the molecule.
chem_shift	chemical shift of the resonance (PPM).
nucleus	nucleus (1H, 31P...).
scale_factor	multiplicative scaling factor.
lw	linewidth in Hz.
lg	Lorentz-Gauss lineshape parameter (between 0 and 1).

Value

mol_parameters object.

get_voi_cog — *Calculate the centre of gravity for an image containing 0 and 1's.*

Description

Calculate the centre of gravity for an image containing 0 and 1's.

Usage

```
get_voi_cog(voi)
```

Arguments

voi	nifti object.

Value

triplet of x,y,z coordinates.

`get_voi_seg` *Return the white matter, gray matter and CSF composition of a volume.*

Description

Return the white matter, gray matter and CSF composition of a volume.

Usage

```
get_voi_seg(voi, mri_seg)
```

Arguments

`voi`	volume data as a nifti object.
`mri_seg`	segmented brain volume as a nifti object.

Value

a vector of partial volumes expressed as percentages.

`get_voi_seg_psf` *Return the white matter, gray matter and CSF composition of a volume.*

Description

Return the white matter, gray matter and CSF composition of a volume.

Usage

```
get_voi_seg_psf(psf, mri_seg)
```

Arguments

`psf`	volume data as a nifti object.
`mri_seg`	segmented brain volume as a nifti object.

Value

a vector of partial volumes expressed as percentages.

get_voxel *Return a single voxel from a larger mrs dataset.*

Description

Return a single voxel from a larger mrs dataset.

Usage

```
get_voxel(mrs_data, x_pos = 1, y_pos = 1, z_pos = 1, dyn = 1, coil = 1)
```

Arguments

mrs_data	MRS data.
x_pos	the x index to plot.
y_pos	the y index to plot.
z_pos	the z index to plot.
dyn	the dynamic index to plot.
coil	the coil element number to plot.

Value

MRS data.

grid_shift_xy *Grid shift MRSI data in the x/y dimension.*

Description

Grid shift MRSI data in the x/y dimension.

Usage

```
grid_shift_xy(mrs_data, x_shift, y_shift)
```

Arguments

mrs_data	MRSI data in the spatial domain.
x_shift	shift to apply in the x-direction in units of voxels.
y_shift	shift to apply in the y-direction in units of voxels.

Value

shifted data.

hsvd_filt *HSVD based signal filter.*

Description

HSVD based signal filter described in: Barkhuijsen H, de Beer R, van Ormondt D. Improved algorithm for noniterative and timedomain model fitting to exponentially damped magnetic resonance signals. J Magn Reson 1987;73:553-557.

Usage

```
hsvd_filt(mrs_data, xlim = c(-30, 30), comps = 40, irlba = TRUE)
```

Arguments

mrs_data	MRS data to be filtered.
xlim	frequency range in Hz to filter.
comps	number of Lorentzian components to use for modelling.
irlba	option to use irlba SVD (logical).

hz *Return the frequency scale of an MRS dataset in Hz.*

Description

Return the frequency scale of an MRS dataset in Hz.

Usage

```
hz(mrs_data, fs = NULL, N = NULL)
```

Arguments

mrs_data	MRS data.
fs	sampling frequency in Hz.
N	number of data points in the spectral dimension.

Value

frequency scale.

ift_shift *Perform an iffshift and ifft on a vector.*

Description

Perform an iffshift and ifft on a vector.

Usage

```
ift_shift(vec_in)
```

Arguments

vec_in	vector input.

Value

output vector.

ift_shift_mat *Perform an ifft and ifftshift on a matrix with each column replaced by its shifted ifft.*

Description

Perform an ifft and ifftshift on a matrix with each column replaced by its shifted ifft.

Usage

```
ift_shift_mat(mat_in)
```

Arguments

mat_in	matrix input.

Value

output matrix.

Im.mrs_data *Apply Im operator to an MRS dataset.*

Description

Apply Im operator to an MRS dataset.

Usage

```
## S3 method for class 'mrs_data'
Im(z)
```

Arguments

z	MRS data.

Value

MRS data following Im operator.

image.mrs_data *Image plot method for objects of class mrs_data.*

Description

Image plot method for objects of class mrs_data.

Usage

```
## S3 method for class 'mrs_data'
image(
  x,
  xlim = NULL,
  mode = "re",
  col = NULL,
  dim = "dyn",
  x_pos = NULL,
  y_pos = NULL,
  z_pos = NULL,
  dyn = 1,
  coil = 1,
  restore_def_par = TRUE,
  y_ticks = NULL,
  vline = NULL,
  hline = NULL,
  ...
)
```

Arguments

x	object of class mrs_data.
xlim	the range of values to display on the x-axis, eg xlim = c(4,1).
mode	representation of the complex numbers to be plotted, can be one of: "re", "im", "mod" or "arg".
col	Colour map to use, defaults to viridis.
dim	the dimension to display on the y-axis, can be one of: "dyn", "x", "y", "z" or "coil".
x_pos	the x index to plot.
y_pos	the y index to plot.
z_pos	the z index to plot.
dyn	the dynamic index to plot.
coil	the coil element number to plot.
restore_def_par	restore default plotting par values after the plot has been made.
y_ticks	a vector of indices specifying where to place tick marks.
vline	draw a vertical line at the value of vline.
hline	draw a horizontal line at the value of hline.
...	other arguments to pass to the plot method.

---

interleave_dyns *Interleave the first and second half of a dynamic series.*

Description

Interleave the first and second half of a dynamic series.

Usage

```
interleave_dyns(mrs_data)
```

Arguments

mrs_data	dynamic MRS data.

Value

interleaved data.

int_spec *Integrate a spectral region.*

Description

Integrate a spectral region.

Usage

```
int_spec(mrs_data, xlim = NULL, scale = "ppm", mode = "re", summation = "sum")
```

Arguments

mrs_data	MRS data.
xlim	spectral range to be integrated (defaults to full range).
scale	units of xlim, can be : "ppm", "Hz" or "points".
mode	spectral mode, can be : "re", "im", "mod" or "cplx".
summation	can be "sum" (default), "mean" or "l2".

Value

an array of integral values.

inv_even_dyns *Invert even numbered dynamic scans starting from 1 (2,4,6...).*

Description

Invert even numbered dynamic scans starting from 1 (2,4,6...).

Usage

```
inv_even_dyns(mrs_data)
```

Arguments

mrs_data	dynamic MRS data.

Value

dynamic MRS data with inverted even numbered scans.

inv_odd_dyns *Invert odd numbered dynamic scans starting from 1 (1,3,5...).*

Description

Invert odd numbered dynamic scans starting from 1 (1,3,5...).

Usage

```
inv_odd_dyns(mrs_data)
```

Arguments

mrs_data	dynamic MRS data.

Value

dynamic MRS data with inverted odd numbered scans.

is_fd *Check if the chemical shift dimension of an MRS data object is in the frequency domain.*

Description

Check if the chemical shift dimension of an MRS data object is in the frequency domain.

Usage

```
is_fd(mrs_data)
```

Arguments

mrs_data	MRS data.

Value

logical value.

l2_reg — *Perform l2 regularisation artefact suppression using the method proposed by Bilgic et al. JMRI 40(1):181-91 2014.*

Description

Perform l2 regularisation artefact suppression using the method proposed by Bilgic et al. JMRI 40(1):181-91 2014.

Usage

```
l2_reg(mrs_data, A, b)
```

Arguments

mrs_data	input data for artefact suppression.
A	matrix of spectral data points containing the artefact basis signals.
b	regularisation parameter.

Value

l2 reconstructed mrs_data object.

lb — *Apply line-broadening (apodisation) to MRS data or basis object.*

Description

Apply line-broadening (apodisation) to MRS data or basis object.

Usage

```
lb(x, lb, lg = 1)

## S3 method for class 'mrs_data'
lb(x, lb, lg = 1)

## S3 method for class 'basis_set'
lb(x, lb, lg = 1)
```

Arguments

x	input mrs_data or basis_set object.
lb	amount of line-broadening in Hz.
lg	Lorentz-Gauss lineshape parameter (between 0 and 1).

Value

line-broadened data.

`lw2alpha` *Covert a linewidth in Hz to an equivalent alpha value in the time-domain ie: x * exp(-t * alpha).*

Description

Covert a linewidth in Hz to an equivalent alpha value in the time-domain ie: x * exp(-t * alpha).

Usage

```
lw2alpha(lw)
```

Arguments

`lw`	linewidth in Hz.

Value

beta damping value.

`lw2beta` *Covert a linewidth in Hz to an equivalent beta value in the time-domain ie: x * exp(-t * t * beta).*

Description

Covert a linewidth in Hz to an equivalent beta value in the time-domain ie: x * exp(-t * t * beta).

Usage

```
lw2beta(lw)
```

Arguments

`lw`	linewidth in Hz.

Value

beta damping value.

mask_xy *Mask an MRSI dataset in the x-y direction*

Description

Mask an MRSI dataset in the x-y direction

Usage

```
mask_xy(mrs_data, x_dim, y_dim)
```

Arguments

mrs_data	MRS data object.
x_dim	x dimension output length.
y_dim	y dimension output length.

Value

masked MRS data.

mask_xy_mat *Mask a 2D MRSI dataset in the x-y dimension.*

Description

Mask a 2D MRSI dataset in the x-y dimension.

Usage

```
mask_xy_mat(mrs_data, mask)
```

Arguments

mrs_data	MRS data object.
mask	matrix of boolean values specifying the voxels to mask, where a value of TRUE indicates the voxel should be removed.

Value

masked dataset.

mat2mrs_data *Convert a matrix (with spectral points in the row dimension and dynamics in the column dimensions) into a mrs_data object.*

Description

Convert a matrix (with spectral points in the row dimension and dynamics in the column dimensions) into a mrs_data object.

Usage

```
mat2mrs_data(mat, fs = def_fs(), ft = def_ft(), ref = def_ref(), fd = FALSE)
```

Arguments

mat	data matrix.
fs	sampling frequency in Hz.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.
fd	flag to indicate if the matrix is in the frequency domain (logical).

Value

mrs_data object.

max_mrs *Apply the max operator to an MRS dataset.*

Description

Apply the max operator to an MRS dataset.

Usage

```
max_mrs(mrs_data)
```

Arguments

mrs_data	MRS data.

Value

MRS data following max operator.

max_mrs_interp *Apply the max operator to an interpolated MRS dataset.*

Description

Apply the max operator to an interpolated MRS dataset.

Usage

```
max_mrs_interp(mrs_data, interp_f = 4)
```

Arguments

mrs_data	MRS data.
interp_f	interpolation factor.

Value

Array of maximum values (real only).

mean.mrs_data *Calculate the mean spectrum from an mrs_data object.*

Description

Calculate the mean spectrum from an mrs_data object.

Usage

```
## S3 method for class 'mrs_data'
mean(x, ...)
```

Arguments

x	object of class mrs_data.
...	other arguments to pass to the colMeans function.

Value

mean mrs_data object.

mean_dyns *Calculate the mean dynamic data.*

Description

Calculate the mean dynamic data.

Usage

```
mean_dyns(mrs_data)
```

Arguments

mrs_data	dynamic MRS data.

Value

mean dynamic data.

mean_dyn_blocks *Calculate the mean of adjacent dynamic scans.*

Description

Calculate the mean of adjacent dynamic scans.

Usage

```
mean_dyn_blocks(mrs_data, block_size)
```

Arguments

mrs_data	dynamic MRS data.
block_size	number of adjacent dynamics scans to average over.

Value

dynamic data averaged in blocks.

mean_dyn_pairs *Calculate the pairwise means across a dynamic data set.*

Description

Calculate the pairwise means across a dynamic data set.

Usage

```
mean_dyn_pairs(mrs_data)
```

Arguments

mrs_data	dynamic MRS data.

Value

mean dynamic data of adjacent dynamic pairs.

median_dyns *Calculate the median dynamic data.*

Description

Calculate the median dynamic data.

Usage

```
median_dyns(mrs_data)
```

Arguments

mrs_data	dynamic MRS data.

Value

median dynamic data.

Mod.mrs_data *Apply Mod operator to an MRS dataset.*

Description

Apply Mod operator to an MRS dataset.

Usage

```
## S3 method for class 'mrs_data'
Mod(z)
```

Arguments

z	MRS data.

Value

MRS data following Mod operator.

mrsi2d_img2kspace *Transform 2D MRSI data to k-space in the x-y direction.*

Description

Transform 2D MRSI data to k-space in the x-y direction.

Usage

```
mrsi2d_img2kspace(mrs_data)
```

Arguments

mrs_data	2D MRSI data.

Value

k-space data.

`mrsi2d_kspace2img` *Transform 2D MRSI data from k-space to image space in the x-y direction.*

Description

Transform 2D MRSI data from k-space to image space in the x-y direction.

Usage

```
mrsi2d_kspace2img(mrs_data)
```

Arguments

`mrs_data`	2D MRSI data.

Value

MRSI data in image space.

`mrs_data2basis` *Convert an mrs_data object to basis object - where basis signals are spread across the dynamic dimension in the MRS data.*

Description

Convert an mrs_data object to basis object - where basis signals are spread across the dynamic dimension in the MRS data.

Usage

```
mrs_data2basis(mrs_data, names)
```

Arguments

`mrs_data`	mrs_data object with basis signals spread across the dynamic dimension.
`names`	list of names corresponding to basis signals.

Value

basis set object.

mrs_data2mat — *Convert mrs_data object to a matrix, with spectral points in the column dimension and dynamics in the row dimension.*

Description

Convert mrs_data object to a matrix, with spectral points in the column dimension and dynamics in the row dimension.

Usage

```
mrs_data2mat(mrs_data)
```

Arguments

mrs_data	MRS data object or list of MRS data objects.

Value

MRS data matrix.

mrs_data2vec — *Convert mrs_data object to a vector.*

Description

Convert mrs_data object to a vector.

Usage

```
mrs_data2vec(mrs_data, dyn = 1, x_pos = 1, y_pos = 1, z_pos = 1, coil = 1)
```

Arguments

mrs_data	MRS data object.
dyn	dynamic index.
x_pos	x index.
y_pos	y index.
z_pos	z index.
coil	coil element index.

Value

MRS data vector.

`mvfftshift` *Perform a fftshift on a matrix, with each column replaced by its shifted result.*

Description

Perform a fftshift on a matrix, with each column replaced by its shifted result.

Usage

```
mvfftshift(x)
```

Arguments

`x`	matrix input.

Value

output matrix.

`mvifftshift` *Perform an ifftshift on a matrix, with each column replaced by its shifted result.*

Description

Perform an ifftshift on a matrix, with each column replaced by its shifted result.

Usage

```
mvifftshift(x)
```

Arguments

`x`	matrix input.

Value

output matrix.

n2coord — *Print fit coordinates from a single index.*

Description

Print fit coordinates from a single index.

Usage

```
n2coord(n, fit_res)
```

Arguments

n	fit index.
fit_res	fit_result object.

Ncoils — *Return the total number of coil elements in an MRS dataset.*

Description

Return the total number of coil elements in an MRS dataset.

Usage

```
Ncoils(mrs_data)
```

Arguments

mrs_data	MRS data.

Ndyns — *Return the total number of dynamic scans in an MRS dataset.*

Description

Return the total number of dynamic scans in an MRS dataset.

Usage

```
Ndyns(mrs_data)
```

Arguments

mrs_data	MRS data.

`nifti_flip_lr` *Flip the x data dimension order of a nifti image. This corresponds to flipping MRI data in the left-right direction, assuming the data in save in neurological format (can check with fslorient program).*

Description

Flip the x data dimension order of a nifti image. This corresponds to flipping MRI data in the left-right direction, assuming the data in save in neurological format (can check with fslorient program).

Usage

```
nifti_flip_lr(x)
```

Arguments

`x`	nifti object to be processed.

Value

nifti object with reversed x data direction.

`norm_mrs` *Normalise mrs_data to a spectral region.*

Description

Normalise mrs_data to a spectral region.

Usage

```
norm_mrs(mrs_data, xlim = NULL, scale = "ppm", mode = "re", summation = "l2")
```

Arguments

`mrs_data`	MRS data.
`xlim`	spectral range to be integrated (defaults to full range).
`scale`	units of xlim, can be : "ppm", "Hz" or "points".
`mode`	spectral mode, can be : "re", "im", "mod" or "cplx".
`summation`	can be "sum", "mean" or "l2" (default).

Value

normalised data.

Npts — *Return the number of data points in an MRS dataset.*

Description

Return the number of data points in an MRS dataset.

Usage

```
Npts(mrs_data)
```

Arguments

mrs_data	MRS data.

Value

number of data points.

Nspec — *Return the total number of spectra in an MRS dataset.*

Description

Return the total number of spectra in an MRS dataset.

Usage

```
Nspec(mrs_data)
```

Arguments

mrs_data	MRS data.

Nx — *Return the total number of x locations in an MRS dataset.*

Description

Return the total number of x locations in an MRS dataset.

Usage

```
Nx(mrs_data)
```

Arguments

mrs_data	MRS data.

Ny — *Return the total number of y locations in an MRS dataset.*

Description

Return the total number of y locations in an MRS dataset.

Usage

```
Ny(mrs_data)
```

Arguments

mrs_data	MRS data.

Nz — *Return the total number of z locations in an MRS dataset.*

Description

Return the total number of z locations in an MRS dataset.

Usage

```
Nz(mrs_data)
```

Arguments

mrs_data	MRS data.

ortho3 — *Display an orthographic projection plot of a nifti object.*

Description

Display an orthographic projection plot of a nifti object.

Usage

```
ortho3(
  underlay,
  overlay = NULL,
  xyz = NULL,
  zlim = NULL,
  zlim_ol = NULL,
  alpha = 1,
  col_ol = viridisLite::viridis(64),
  orient_lab = TRUE,
  rescale = 1,
  crosshairs = TRUE,
  ch_lwd = 1,
  colourbar = TRUE,
  bg = "black",
  mar = c(0, 0, 0, 0),
  smallplot = c(0.5, 0.52, 0.1, 0.4)
)
```

Arguments

underlay	underlay image to be shown in grayscale.
overlay	optional overlay image.
xyz	x, y, z slice coordinates to display.
zlim	underlay intensity limits.
zlim_ol	overlay intensity limits.
alpha	transparency of overlay.
col_ol	colour palette of overlay.
orient_lab	display orientation labels (default TRUE).
rescale	rescale factor for the underlay and overlay images.
crosshairs	display the crosshairs (default TRUE).
ch_lwd	crosshair linewidth.
colourbar	display a colourbar for the overlay (default TRUE).
bg	plot background colour.
mar	plot margins.
smallplot	smallplot option for positioning the colourbar.

ortho3_int — *Display an interactive orthographic projection plot of a nifti object.*

Description

Display an interactive orthographic projection plot of a nifti object.

Usage

```
ortho3_int(
  underlay,
  overlay = NULL,
  xyz = NULL,
  zlim = NULL,
  zlim_ol = NULL,
  alpha = 1,
  ...
)
```

Arguments

underlay	underlay image to be shown in grayscale.
overlay	optional overlay image.
xyz	x, y, z slice coordinates to display.
zlim	underlay intensity limits.
zlim_ol	overlay intensity limits.
alpha	transparency of overlay.
...	other options to be passed to the ortho3 function.

peak_info — *Search for the highest peak in a spectral region and return the frequency, height and FWHM.*

Description

Search for the highest peak in a spectral region and return the frequency, height and FWHM.

Usage

```
peak_info(
  mrs_data,
  xlim = c(4, 0.5),
  interp_f = 4,
  scale = "ppm",
  mode = "real"
)
```

Arguments

mrs_data	an object of class mrs_data.
xlim	frequency range (default units of PPM) to search for the highest peak.
interp_f	interpolation factor, defaults to 4x.
scale	the units to use for the frequency scale, can be one of: "ppm", "hz" or "points".
mode	spectral mode, can be : "real", "imag" or "mod".

Value

list of arrays containing the highest peak frequency, height and FWHM in units of PPM and Hz.

---

phase *Apply phasing parameters to MRS data.*

Description

Apply phasing parameters to MRS data.

Usage

```
phase(mrs_data, zero_order, first_order = 0)
```

Arguments

mrs_data	MRS data.
zero_order	zero'th order phase term in degrees.
first_order	first order (frequency dependent) phase term in ms.

Value

MRS data with applied phase parameters.

plot.fit_result *Plot the fitting results of an object of class* fit_result.

Description

Plot the fitting results of an object of class fit_result.

Usage

```
## S3 method for class 'fit_result'
plot(
  x,
  dyn = 1,
  x_pos = 1,
  y_pos = 1,
  z_pos = 1,
  coil = 1,
  xlim = NULL,
  data_only = FALSE,
  label = NULL,
  plot_sigs = NULL,
  n = NULL,
  sub_bl = FALSE,
  mar = NULL,
  restore_def_par = TRUE,
  ylim = NULL,
  y_scale = FALSE,
  ...
)
```

Arguments

x	fit_result object.
dyn	the dynamic index to plot.
x_pos	the x index to plot.
y_pos	the y index to plot.
z_pos	the z index to plot.
coil	the coil element number to plot.
xlim	the range of values to display on the x-axis, eg xlim = c(4,1).
data_only	display only the processed data (logical).
label	character string to add to the top left of the plot window.
plot_sigs	a character vector of signal names to add to the plot.
n	single index element to plot (overrides other indices when given).
sub_bl	subtract the baseline from the data and fit (logical).

mar	option to adjust the plot margins. See ?par.
restore_def_par	restore default plotting par values after the plot has been made.
ylim	range of values to display on the y-axis, eg ylim = c(0,10).
y_scale	option to display the y-axis values (logical).
...	further arguments to plot method.

`plot.mrs_data` *Plotting method for objects of class mrs_data.*

Description

Plotting method for objects of class mrs_data.

Usage

```
## S3 method for class 'mrs_data'
plot(
  x,
  dyn = 1,
  x_pos = 1,
  y_pos = 1,
  z_pos = 1,
  coil = 1,
  fd = TRUE,
  x_units = NULL,
  xlim = NULL,
  y_scale = FALSE,
  x_ax = TRUE,
  mode = "re",
  lwd = NULL,
  bty = NULL,
  label = "",
  restore_def_par = TRUE,
  mar = NULL,
  xaxis_lab = NULL,
  ...
)
```

Arguments

x	object of class mrs_data.
dyn	the dynamic index to plot.
x_pos	the x index to plot.
y_pos	the y index to plot.

z_pos	the z index to plot.
coil	the coil element number to plot.
fd	display data in the frequency-domain (default), or time-domain (logical).
x_units	the units to use for the x-axis, can be one of: "ppm", "hz", "points" or "seconds".
xlim	the range of values to display on the x-axis, eg xlim = c(4,1).
y_scale	option to display the y-axis values (logical).
x_ax	option to display the x-axis values (logical).
mode	representation of the complex numbers to be plotted, can be one of: "re", "im", "mod" or "arg".
lwd	plot linewidth.
bty	option to draw a box around the plot. See ?par.
label	character string to add to the top left of the plot window.
restore_def_par	restore default plotting par values after the plot has been made.
mar	option to adjust the plot margins. See ?par.
xaxis_lab	x-axis label.
...	other arguments to pass to the plot method.

`plot_bc` *Convenience function to plot a baseline estimate with the original data.*

Description

Convenience function to plot a baseline estimate with the original data.

Usage

```
plot_bc(orig_data, bc_data, ...)
```

Arguments

orig_data	the original data.
bc_data	the baseline corrected data.
...	other arguments to pass to the stackplot function.

plot_slice_fit *Plot a 2D slice from an MRSI fit result object.*

Description

Plot a 2D slice from an MRSI fit result object.

Usage

```
plot_slice_fit(
  fit_res,
  map,
  map_denom = NULL,
  slice = 1,
  zlim = NULL,
  interp = 1
)
```

Arguments

fit_res	fit_result object.
map	fit result values to display as a colour map. Can be specified as a character string or array of numeric values. Defaults to "tNAA".
map_denom	fit result values to divide the map argument by. Can be specified as a character string (eg "tCr") or array of numeric values.
slice	slice to plot in the z direction.
zlim	range of values to plot.
interp	interpolation factor.

plot_slice_fit_inter *Plot a 2D slice from an MRSI fit result object.*

Description

Plot a 2D slice from an MRSI fit result object.

Usage

```
plot_slice_fit_inter(
  fit_res,
  map = NULL,
  map_denom = NULL,
  slice = 1,
  zlim = NULL,
  interp = 1,
  xlim = NULL
)
```

Arguments

fit_res	fit_result object.
map	fit result values to display as a colour map. Can be specified as a character string or array of numeric values. Defaults to "tNAA".
map_denom	fit result values to divide the map argument by. Can be specified as a character string (eg "tCr") or array of numeric values.
slice	slice to plot in the z direction.
zlim	range of values to plot.
interp	interpolation factor.
xlim	spectral plot limits for the x axis.

---

plot_slice_map *Plot a slice from a 7 dimensional array.*

Description

Plot a slice from a 7 dimensional array.

Usage

```
plot_slice_map(
  data,
  zlim = NULL,
  mask_map = NULL,
  mask_cutoff = 20,
  interp = 1,
  slice = 1,
  dyn = 1,
  coil = 1,
  ref = 1,
  denom = NULL,
  horizontal = FALSE
)
```

Arguments

data	7d array of values to be plotted.
zlim	smallest and largest values to be plotted.
mask_map	matching map with logical values to indicate if the corresponding values should be plotted.
mask_cutoff	minimum values to plot (as a percentage of the maximum).
interp	map interpolation factor.
slice	the slice index to plot.

dyn	the dynamic index to plot.
coil	the coil element number to plot.
ref	reference index to plot.
denom	map to use as a denominator.
horizontal	display the colourbar horizontally (logical).

`plot_slice_map_inter` *Plot an interactive slice map from a data array where voxels can be selected to display a corresponding spectrum.*

Description

Plot an interactive slice map from a data array where voxels can be selected to display a corresponding spectrum.

Usage

```
plot_slice_map_inter(
  mrs_data,
  map = NULL,
  xlim = NULL,
  slice = 1,
  zlim = NULL,
  mask_map = NULL,
  denom = NULL,
  mask_cutoff = 20,
  interp = 1,
  mode = "re",
  y_scale = FALSE,
  ylim = NULL,
  coil = 1
)
```

Arguments

mrs_data	spectral data.
map	array of values to be plotted, defaults to the integration of the modulus of the full spectral width.
xlim	spectral region to plot.
slice	the slice index to plot.
zlim	smallest and largest values to be plotted.
mask_map	matching map with logical values to indicate if the corresponding values should be plotted.
denom	map to use as a denominator.

mask_cutoff	minimum values to plot (as a percentage of the maximum).
interp	map interpolation factor.
mode	representation of the complex spectrum to be plotted, can be one of: "re", "im", "mod" or "arg".
y_scale	option to display the y-axis values (logical).
ylim	intensity range to plot.
coil	coil element to plot.

---

plot_voi_overlay *Plot a volume as an image overlay.*

Description

Plot a volume as an image overlay.

Usage

```
plot_voi_overlay(voi, mri, flip_lr = TRUE)
```

Arguments

voi	volume data as a nifti object.
mri	image data as a nifti object.
flip_lr	flip the image in the left-right direction.

---

plot_voi_overlay_seg *Plot a volume as an overlay on a segmented brain volume.*

Description

Plot a volume as an overlay on a segmented brain volume.

Usage

```
plot_voi_overlay_seg(voi, mri_seg, flip_lr = TRUE)
```

Arguments

voi	volume data as a nifti object.
mri_seg	segmented brain volume as a nifti object.
flip_lr	flip the image in the left-right direction.

ppm — *Return the ppm scale of an MRS dataset or fit result.*

Description

Return the ppm scale of an MRS dataset or fit result.

Usage

```
ppm(x, ft = NULL, ref = NULL, fs = NULL, N = NULL)

## S3 method for class 'mrs_data'
ppm(x, ft = NULL, ref = NULL, fs = NULL, N = NULL)

## S3 method for class 'fit_result'
ppm(x, ft = NULL, ref = NULL, fs = NULL, N = NULL)
```

Arguments

x	MRS dataset of fit result.
ft	transmitter frequency in Hz, does not apply when the object is a fit result.
ref	reference value for ppm scale, does not apply when the object is a fit result.
fs	sampling frequency in Hz, does not apply when the object is a fit result.
N	number of data points in the spectral dimension, does not apply when the object is a fit result.

Value

ppm scale.

print.fit_result — *Print a summary of an object of class* fit_result.

Description

Print a summary of an object of class fit_result.

Usage

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

Arguments

x	fit_result object.
...	further arguments.

print.mrs_data *Print a summary of mrs_data parameters.*

Description

Print a summary of mrs_data parameters.

Usage

```
## S3 method for class 'mrs_data'
print(x, full = FALSE, ...)
```

Arguments

x	mrs_data object.
full	print all parameters (default FALSE).
...	further arguments.

qn_states *Get the quantum coherence matrix for a spin system.*

Description

Get the quantum coherence matrix for a spin system.

Usage

```
qn_states(sys)
```

Arguments

sys	spin system object.

Value

quantum coherence number matrix.

rats *Robust Alignment to a Target Spectrum (RATS).*

Description

Robust Alignment to a Target Spectrum (RATS).

Usage

```
rats(
  mrs_data,
  ref = NULL,
  xlim = c(4, 0.5),
  max_shift = 20,
  p_deg = 2,
  max_t = 0.2
)
```

Arguments

mrs_data	MRS data to be corrected.
ref	optional MRS data to use as a reference, the mean of all dynamics is used if this argument is not supplied.
xlim	optional frequency range to perform optimisation, set to NULL to use the full range.
max_shift	maximum allowable frequency shift in Hz.
p_deg	polynomial degree used for baseline modelling. Negative values disable baseline modelling.
max_t	truncate the FID when longer than max_t to reduce time taken

Value

a list containing the corrected data; phase and shift values in units of degrees and Hz respectively.

Re.mrs_data *Apply Re operator to an MRS dataset.*

Description

Apply Re operator to an MRS dataset.

Usage

```
## S3 method for class 'mrs_data'
Re(z)
```

Arguments

z	MRS data.

Value

MRS data following Re operator.

---

read_basis *Read a basis file in LCModel .basis format.*

Description

Read a basis file in LCModel .basis format.

Usage

```
read_basis(basis_file, ref = def_ref())
```

Arguments

basis_file	path to basis file.
ref	assumed ppm reference value.

Value

basis object.

---

read_ima_coil_dir *Read a directory containing Siemens MRS IMA files and combine along the coil dimension. Note that the coil ID is inferred from the sorted file name and should be checked when consistency is required between two directories.*

Description

Read a directory containing Siemens MRS IMA files and combine along the coil dimension. Note that the coil ID is inferred from the sorted file name and should be checked when consistency is required between two directories.

Usage

```
read_ima_coil_dir(dir)
```

Arguments

dir	data directory path.

Value

mrs_data object.

`read_ima_dyn_dir` *Read a directory containing Siemens MRS IMA files and combine along the dynamic dimension. Note that the coil ID is inferred from the sorted file name and should be checked when consistency is required.*

Description

Read a directory containing Siemens MRS IMA files and combine along the dynamic dimension. Note that the coil ID is inferred from the sorted file name and should be checked when consistency is required.

Usage

```
read_ima_dyn_dir(dir)
```

Arguments

`dir`	data directory path.

Value

mrs_data object.

`read_lcm_coord` *Read an LCModel formatted coord file containing fit information.*

Description

Read an LCModel formatted coord file containing fit information.

Usage

```
read_lcm_coord(coord_f)
```

Arguments

`coord_f`	path to the coord file.

Value

list containing a table of fit point and results structure containing signal amplitudes, errors and fitting diagnostics.

read_mrs *Read MRS data from a file.*

Description

Read MRS data from a file.

Usage

```
read_mrs(
  fname,
  format = NULL,
  ft = NULL,
  fs = NULL,
  ref = NULL,
  n_ref_scans = NULL,
  full_data = FALSE,
  verbose = FALSE
)
```

Arguments

fname	filename of the dpt format MRS data.
format	string describing the data format. Must be one of the following : "spar_sdat", "rda", "ima", "twix", "pfile", "list_data", "paravis", "dpt", "lcm_raw", "rds", "nifti". If not specified, the format will be guessed from the filename extension.
ft	transmitter frequency in Hz (required for list_data format).
fs	sampling frequency in Hz (required for list_data format).
ref	reference value for ppm scale (required for list_data format).
n_ref_scans	override the number of water reference scans detected in the file header (GE p-file only).
full_data	export all data points, including those before the start of the FID (default = FALSE).
verbose	print data file information (default = FALSE).

Value

MRS data object.

Examples

```
fname <- system.file("extdata", "philips_spar_sdat_WS.SDAT", package = "spant")
mrs_data <- read_mrs(fname)
print(mrs_data)
```

read_mrs_dpt *Read MRS data stored in dangerplot (dpt) v3 format.*

Description

Read MRS data stored in dangerplot (dpt) v3 format.

Usage

```
read_mrs_dpt(fname)
```

Arguments

fname	filename of the dpt format MRS data.

Value

MRS data object.

Examples

```
## Not run:
mrs_data <- read_mrs_dpt(system.file("extdata","svs.dpt",package="spant"))

## End(Not run)
```

read_mrs_tqn *Read MRS data using the TARQUIN software package.*

Description

Read MRS data using the TARQUIN software package.

Usage

```
read_mrs_tqn(fname, fname_ref = NA, format, id = NA, group = NA)
```

Arguments

fname	the filename containing the MRS data.
fname_ref	a second filename containing reference MRS data.
format	format of the MRS data. Can be one of the following: siemens, philips, ge, dcm, dpt, rda, lcm, varian, bruker, jmrui_txt.
id	optional ID string.
group	optional group string.

Value

MRS data object.

Examples

```
fname <- system.file("extdata","philips_spar_sdat_WS.SDAT",package="spant")
## Not run:
mrs_data <- read_mrs_tqn(fname, format="philips")

## End(Not run)
```

`read_siemens_txt_hdr` *Read the text format header found in Siemens IMA and TWIW data files.*

Description

Read the text format header found in Siemens IMA and TWIW data files.

Usage

```
read_siemens_txt_hdr(fname, version = "vd")
```

Arguments

`fname`	file name to read.
`version`	software version, can be "vb" or "vd".

Value

a list of parameter values

`read_tqn_fit` *Reader for csv fit results generated by TARQUIN.*

Description

Reader for csv fit results generated by TARQUIN.

Usage

```
read_tqn_fit(fit_f)
```

Arguments

`fit_f`	TARQUIN fit file.

Value

A data frame of the fit data points.

Examples

```
## Not run:
fit <- read_tqn_fit(system.file("extdata","fit.csv",package="spant"))

## End(Not run)
```

`read_tqn_result` *Reader for csv results generated by TARQUIN.*

Description

Reader for csv results generated by TARQUIN.

Usage

```
read_tqn_result(result_f, remove_rcs = TRUE)
```

Arguments

`result_f`	TARQUIN result file.
`remove_rcs`	omit row, column and slice ids from output.

Value

list of amplitudes, crlbs and diagnostics.

Examples

```
## Not run:
result <- read_tqn_result(system.file("extdata","result.csv",package="spant"))

## End(Not run)
```

rep_array_dim *Repeat an array over a given dimension.*

Description

Repeat an array over a given dimension.

Usage

```
rep_array_dim(x, rep_dim, n)
```

Arguments

x	array.
rep_dim	dimension to extend.
n	number of times to repeat.

Value

extended array.

rep_dyn *Replicate a scan in the dynamic dimension.*

Description

Replicate a scan in the dynamic dimension.

Usage

```
rep_dyn(mrs_data, times)
```

Arguments

mrs_data	MRS data to be replicated.
times	number of times to replicate.

Value

replicated data object.

rep_mrs *Replicate a scan over a given dimension.*

Description

Replicate a scan over a given dimension.

Usage

```
rep_mrs(mrs_data, x_rep = 1, y_rep = 1, z_rep = 1, dyn_rep = 1, coil_rep = 1)
```

Arguments

mrs_data	MRS data to be replicated.
x_rep	number of x replications.
y_rep	number of y replications.
z_rep	number of z replications.
dyn_rep	number of dynamic replications.
coil_rep	number of coil replications.

Value

replicated data object.

resample_img *Resample an image to match a target image space.*

Description

Resample an image to match a target image space.

Usage

```
resample_img(source, target, interp = 3L)
```

Arguments

source	image data as a nifti object.
target	image data as a nifti object.
interp	interpolation parameter, see nifyreg.linear definition.

Value

resampled image data as a nifti object.

`resample_voi` *Resample a VOI to match a target image space using nearest-neighbour interpolation.*

Description

Resample a VOI to match a target image space using nearest-neighbour interpolation.

Usage

```
resample_voi(voi, mri)
```

Arguments

`voi`	volume data as a nifti object.
`mri`	image data as a nifti object.

Value

volume data as a nifti object.

`reslice_to_mrs` *Reslice a nifti object to match the orientation of mrs data.*

Description

Reslice a nifti object to match the orientation of mrs data.

Usage

```
reslice_to_mrs(mri, mrs)
```

Arguments

`mri`	nifti object to be resliced.
`mrs`	mrs_data object for the target orientation.

Value

resliced imaging data.

re_weighting *Apply a weighting to the FID to enhance spectral resolution.*

Description

Apply a weighting to the FID to enhance spectral resolution.

Usage

```
re_weighting(mrs_data, re, alpha)
```

Arguments

mrs_data	data to be enhanced.
re	resolution enhancement factor (rising exponential factor).
alpha	alpha factor (Guassian decay)

Value

resolution enhanced mrs_data.

rm_dyns *Remove a subset of dynamic scans.*

Description

Remove a subset of dynamic scans.

Usage

```
rm_dyns(mrs_data, subset)
```

Arguments

mrs_data	dynamic MRS data.
subset	vector containing indices to the dynamic scans to be removed.

Value

MRS data without the specified dynamic scans.

scale_amp_molal_pvc *Apply partial volume correction to a fitting result object.*

Description

Apply partial volume correction to a fitting result object.

Usage

```
scale_amp_molal_pvc(fit_result, ref_data, p_vols, te, tr)
```

Arguments

fit_result	result object generated from fitting.
ref_data	water reference MRS data object.
p_vols	a numeric vector of partial volumes.
te	the MRS TE.
tr	the MRS TR.

Value

A fit_result object with a rescaled results table.

scale_amp_molar *Apply water reference scaling to a fitting results object to yield metabolite quantities in millimolar (mM) units (mol/litre).*

Description

Apply water reference scaling to a fitting results object to yield metabolite quantities in millimolar (mM) units (mol/litre).

Usage

```
scale_amp_molar(fit_result, ref_data, w_att = 0.7, w_conc = 35880)
```

Arguments

fit_result	a result object generated from fitting.
ref_data	water reference MRS data object.
w_att	water attenuation factor (default = 0.7).
w_conc	assumed water concentration (default = 35880).

Value

a fit_result object with a rescaled results table.

scale_amp_ratio *Scale fitted amplitudes to a ratio of signal amplitude.*

Description

Scale fitted amplitudes to a ratio of signal amplitude.

Usage

```
scale_amp_ratio(fit_result, name)
```

Arguments

fit_result	a result object generated from fitting.
name	the signal name to use as a denominator (usually, "tCr" or "tNAA").

Value

a fit_result object with a rescaled results table.

scale_amp_water_ratio *Scale metabolite amplitudes as a ratio to the unsuppressed water amplitude.*

Description

Scale metabolite amplitudes as a ratio to the unsuppressed water amplitude.

Usage

```
scale_amp_water_ratio(fit_result, ref_data)
```

Arguments

fit_result	a result object generated from fitting.
ref_data	a water reference MRS data object.

Value

a fit_result object with a rescaled results table.

sd — *Calculate the standard deviation spectrum from an mrs_data object.*

Description

Calculate the standard deviation spectrum from an mrs_data object.

Usage

```
sd(x, na.rm)
```

Arguments

x	object of class mrs_data.
na.rm	remove NA values.

Value

sd mrs_data object.

sd.mrs_data — *Calculate the standard deviation spectrum from an mrs_data object.*

Description

Calculate the standard deviation spectrum from an mrs_data object.

Usage

```
## S3 method for class 'mrs_data'
sd(x, na.rm = FALSE)
```

Arguments

x	object of class mrs_data.
na.rm	remove NA values.

Value

sd mrs_data object.

seconds *Return a time scale vector to match the FID of an MRS data object.*

Description

Return a time scale vector to match the FID of an MRS data object.

Usage

```
seconds(mrs_data)
```

Arguments

mrs_data	MRS data.

Value

time scale vector in units of seconds.

seq_cpmg_ideal *CPMG style sequence with ideal pulses.*

Description

CPMG style sequence with ideal pulses.

Usage

```
seq_cpmg_ideal(spin_params, ft, ref, TE = 0.03, echoes = 4)
```

Arguments

spin_params	spin system definition.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.
TE	echo time in seconds.
echoes	number of echoes.

Value

list of resonance amplitudes and frequencies.

`seq_mega_press_ideal` *MEGA-PRESS sequence with ideal localisation pulses and Gaussian shaped editing pulse.*

Description

MEGA-PRESS sequence with ideal localisation pulses and Gaussian shaped editing pulse.

Usage

```
seq_mega_press_ideal(
  spin_params,
  ft,
  ref,
  ed_freq = 1.89,
  TE1 = 0.015,
  TE2 = 0.053,
  BW = 110,
  steps = 50
)
```

Arguments

`spin_params`	spin system definition.
`ft`	transmitter frequency in Hz.
`ref`	reference value for ppm scale.
`ed_freq`	editing pulse frequency in ppm.
`TE1`	TE1 sequence parameter in seconds (TE=TE1+TE2).
`TE2`	TE2 sequence parameter in seconds.
`BW`	editing pulse bandwidth in Hz.
`steps`	number of hard pulses used to approximate the editing pulse.

Value

list of resonance amplitudes and frequencies.

seq_press_ideal *PRESS sequence with ideal pulses.*

Description

PRESS sequence with ideal pulses.

Usage

```
seq_press_ideal(spin_params, ft, ref, TE1 = 0.01, TE2 = 0.02)
```

Arguments

spin_params	spin system definition.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.
TE1	TE1 sequence parameter in seconds (TE=TE1+TE2).
TE2	TE2 sequence parameter in seconds.

Value

list of resonance amplitudes and frequencies.

seq_pulse_acquire *Simple pulse and acquire sequence with ideal pulses.*

Description

Simple pulse and acquire sequence with ideal pulses.

Usage

```
seq_pulse_acquire(spin_params, ft, ref)
```

Arguments

spin_params	spin system definition.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.

Value

list of resonance amplitudes and frequencies.

seq_pulse_acquire_31p *Simple pulse and acquire sequence with ideal pulses.*

Description

Simple pulse and acquire sequence with ideal pulses.

Usage

```
seq_pulse_acquire_31p(spin_params, ft, ref)
```

Arguments

spin_params	spin system definition.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.

Value

list of resonance amplitudes and frequencies.

seq_slaser_ideal *sLASER sequence with ideal pulses.*

Description

sLASER sequence with ideal pulses.

Usage

```
seq_slaser_ideal(spin_params, ft, ref, TE1 = 0.008, TE2 = 0.011, TE3 = 0.009)
```

Arguments

spin_params	spin system definition.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.
TE1	first echo time (between exc. and 1st echo) in seconds.
TE2	second echo time (between 2nd echo and 4th echo) in seconds.
TE3	third echo time (between 4th echo and 5th echo) in seconds.

Value

list of resonance amplitudes and frequencies.

seq_spin_echo_ideal *Spin echo sequence with ideal pulses.*

Description

Spin echo sequence with ideal pulses.

Usage

```
seq_spin_echo_ideal(spin_params, ft, ref, TE = 0.03)
```

Arguments

spin_params	spin system definition.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.
TE	echo time in seconds.

Value

list of resonance amplitudes and frequencies.

seq_spin_echo_ideal_31p *Spin echo sequence with ideal pulses.*

Description

Spin echo sequence with ideal pulses.

Usage

```
seq_spin_echo_ideal_31p(spin_params, ft, ref, TE = 0.03)
```

Arguments

spin_params	spin system definition.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.
TE	echo time in seconds.

Value

list of resonance amplitudes and frequencies.

seq_steam_ideal *STEAM sequence with ideal pulses.*

Description

STEAM sequence with ideal pulses.

Usage

```
seq_steam_ideal(spin_params, ft, ref, TE = 0.03, TM = 0.02)
```

Arguments

spin_params	spin system definition.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.
TE	sequence parameter in seconds.
TM	sequence parameter in seconds.

Value

list of resonance amplitudes and frequencies.

set_def_acq_paras *Set the default acquisition parameters.*

Description

Set the default acquisition parameters.

Usage

```
set_def_acq_paras(
  ft = getOption("spant.def_ft"),
  fs = getOption("spant.def_fs"),
  N = getOption("spant.def_N"),
  ref = getOption("spant.def_ref")
)
```

Arguments

ft	transmitter frequency in Hz.
fs	sampling frequency in Hz.
N	number of data points in the spectral dimension.
ref	reference value for ppm scale.

set_lcm_cmd *Set the command to run the LCModel command-line program.*

Description

Set the command to run the LCModel command-line program.

Usage

```
set_lcm_cmd(cmd)
```

Arguments

cmd	path to binary.

set_lw *Apply line-broadening to an mrs_data object to achieve a specified linewidth.*

Description

Apply line-broadening to an mrs_data object to achieve a specified linewidth.

Usage

```
set_lw(mrs_data, lw, xlim = c(4, 0.5))
```

Arguments

mrs_data	data in.
lw	target linewidth in units of ppm.
xlim	region to search for peaks to obtain a linewidth estimate.

Value

line-broadened data.

set_ref *Set the ppm reference value (eg ppm value at 0Hz).*

Description

Set the ppm reference value (eg ppm value at 0Hz).

Usage

```
set_ref(mrs_data, ref)
```

Arguments

mrs_data	MRS data.
ref	reference value for ppm scale.

set_td_pts *Set the number of time-domain data points, truncating or zero-filling as appropriate.*

Description

Set the number of time-domain data points, truncating or zero-filling as appropriate.

Usage

```
set_td_pts(mrs_data, pts)
```

Arguments

mrs_data	MRS data.
pts	number of data points.

Value

MRS data with pts data points.

`set_tqn_cmd` *Set the command to run the TARQUIN command-line program.*

Description

Set the command to run the TARQUIN command-line program.

Usage

```
set_tqn_cmd(cmd)
```

Arguments

`cmd`	path to binary.

`shift` *Apply a frequency shift to MRS data.*

Description

Apply a frequency shift to MRS data.

Usage

```
shift(mrs_data, shift, units = "ppm")
```

Arguments

`mrs_data`	MRS data.
`shift`	frequency shift (in ppm by default).
`units`	of the shift ("ppm" or "hz").

Value

frequency shifted MRS data.

sim_basis *Simulate a basis set object.*

Description

Simulate a basis set object.

Usage

```
sim_basis(
  mol_list,
  pul_seq = seq_pulse_acquire,
  acq_paras = def_acq_paras(),
  xlim = NULL,
  ...
)
```

Arguments

mol_list	list of mol_parameter objects.
pul_seq	pulse sequence function to use.
acq_paras	list of acquisition parameters or an mrs_data object. See def_acq_paras
xlim	ppm range limiting signals to be simulated.
...	extra parameters to pass to the pulse sequence function.

Value

basis object.

sim_basis_1h_brain *Simulate a basis-set suitable for 1H brain MRS analysis acquired with a PRESS sequence. Note, ideal pulses are assumed.*

Description

Simulate a basis-set suitable for 1H brain MRS analysis acquired with a PRESS sequence. Note, ideal pulses are assumed.

Usage

```
sim_basis_1h_brain(
  pul_seq = seq_press_ideal,
  acq_paras = def_acq_paras(),
  xlim = c(0.5, 4.2),
  lcm_compat = FALSE,
  ...
)
```

Arguments

pul_seq	pulse sequence function to use.
acq_paras	list of acquisition parameters or an mrs_data object. See def_acq_paras.
xlim	range of frequencies to simulate in ppm.
lcm_compat	exclude lipid and MM signals for use with default LCModel options.
...	extra parameters to pass to the pulse sequence function.

Value

basis object.

---

sim_basis_1h_brain_press

Simulate a basis-set suitable for 1H brain MRS analysis acquired with a PRESS sequence. Note, ideal pulses are assumed.

Description

Simulate a basis-set suitable for 1H brain MRS analysis acquired with a PRESS sequence. Note, ideal pulses are assumed.

Usage

```
sim_basis_1h_brain_press(
  acq_paras = def_acq_paras(),
  xlim = c(0.5, 4.2),
  lcm_compat = FALSE,
  TE1 = 0.01,
  TE2 = 0.02
)
```

Arguments

acq_paras	list of acquisition parameters or an mrs_data object. See def_acq_paras
xlim	range of frequencies to simulate in ppm.
lcm_compat	exclude lipid and MM signals for use with default LCModel options.
TE1	TE1 of PRESS sequence (TE = TE1 + TE2).
TE2	TE2 of PRESS sequence.

Value

basis object.

sim_basis_tqn *Simulate a basis file using TARQUIN.*

Description

Simulate a basis file using TARQUIN.

Usage

```
sim_basis_tqn(
  fs = def_fs(),
  ft = def_ft(),
  N = def_N(),
  ref = def_ref(),
  opts = NULL
)
```

Arguments

fs	sampling frequency
ft	transmitter frequency
N	number of data points
ref	chemical shift reference
opts	list of options to pass to TARQUIN.

Examples

```
## Not run:
write_basis_tqn('test.basis',mrs_data,c("--echo","0.04"))

## End(Not run)
```

sim_brain_1h *Simulate MRS data with a similar appearance to normal brain (by default).*

Description

Simulate MRS data with a similar appearance to normal brain (by default).

Usage

```
sim_brain_1h(
  acq_paras = def_acq_paras(),
  type = "normal_v1",
  pul_seq = seq_press_ideal,
  xlim = c(0.5, 4.2),
  full_output = FALSE,
  amps = NULL,
  ...
)
```

Arguments

acq_paras	list of acquisition parameters or an mrs_data object. See def_acq_paras.
type	type of spectrum, only "normal" is implemented currently.
pul_seq	pulse sequence function to use.
xlim	range of frequencies to simulate in ppm.
full_output	when FALSE (default) only output the simulated MRS data. When TRUE output a list containing the MRS data, basis set object and corresponding amplitudes.
amps	a vector of basis amplitudes may be specified to modify the output spectrum.
...	extra parameters to pass to the pulse sequence function.

Value

see full_output option.

---

sim_mol *Simulate a* mol_parameter *object.*

Description

Simulate a mol_parameter object.

Usage

```
sim_mol(
  mol,
  pul_seq = seq_pulse_acquire,
  ft = def_ft(),
  ref = def_ref(),
  fs = def_fs(),
  N = def_N(),
  xlim = NULL,
  ...
)
```

Arguments

mol	mol_parameter object.
pul_seq	pulse sequence function to use.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.
fs	sampling frequency in Hz.
N	number of data points in the spectral dimension.
xlim	ppm range limiting signals to be simulated.
...	extra parameters to pass to the pulse sequence function.

Value

mrs_data object.

---

sim_noise *Simulate a time-domain mrs_data object containing simulated Gaussian noise.*

Description

Simulate a time-domain mrs_data object containing simulated Gaussian noise.

Usage

```
sim_noise(
  sd = 0.1,
  fs = def_fs(),
  ft = def_ft(),
  N = def_N(),
  ref = def_ref(),
  dyns = 1,
  fd = TRUE
)
```

Arguments

sd	standard deviation of the noise.
fs	sampling frequency in Hz.
ft	transmitter frequency in Hz.
N	number of data points in the spectral dimension.
ref	reference value for ppm scale.
dyns	number of dynamic scans to generate.
fd	return data in the frequency-domain (TRUE) or time-domain (FALSE)

Value

mrs_data object.

`sim_resonances` *Simulate a MRS data object containing a set of simulated resonances.*

Description

Simulate a MRS data object containing a set of simulated resonances.

Usage

```
sim_resonances(
  freq = 0,
  amp = 1,
  lw = 0,
  lg = 0,
  phase = 0,
  freq_ppm = TRUE,
  acq_paras = def_acq_paras()
)
```

Arguments

`freq`	resonance frequency.
`amp`	resonance amplitude.
`lw`	line width in Hz.
`lg`	Lorentz-Gauss lineshape parameter (between 0 and 1).
`phase`	phase in degrees.
`freq_ppm`	frequencies are given in ppm units if set to TRUE, otherwise Hz are assumed.
`acq_paras`	list of acquisition parameters. See `def_acq_paras`

Value

MRS data object.

Examples

```
sim_data <- sim_resonances(freq = 2, lw = 5)
```

spant_mpress_drift *Example MEGA-PRESS data with significant B0 drift.*

Description

Example MEGA-PRESS data with significant B0 drift.

Usage

```
spant_mpress_drift
```

Format

An object of class `mrs_data` of length 13.

spin_sys *Create a spin system object for pulse sequence simulation.*

Description

Create a spin system object for pulse sequence simulation.

Usage

```
spin_sys(spin_params, ft, ref)
```

Arguments

`spin_params`	an object describing the spin system properties.
`ft`	transmitter frequency in Hz.
`ref`	reference value for ppm scale.

Value

spin system object.

spm_pve2categorical *Convert SPM style segmentation files to a single categorical image where the numerical values map as: 0) Other, 1) CSF, 2) GM and 3) WM.*

Description

Convert SPM style segmentation files to a single categorical image where the numerical values map as: 0) Other, 1) CSF, 2) GM and 3) WM.

Usage

```
spm_pve2categorical(fname)
```

Arguments

fname	any of the segmentation files (eg c1_MY_T1.nii).

Value

nifti object.

stackplot *Produce a plot with multiple traces.*

Description

Produce a plot with multiple traces.

Usage

```
stackplot(x, ...)
```

Arguments

x	object for plotting.
...	arguments to be passed to methods.

`stackplot.fit_result` *Plot the fitting results of an object of class* `fit_result` *with individual basis set components shown.*

Description

Plot the fitting results of an object of class `fit_result` with individual basis set components shown.

Usage

```
## S3 method for class 'fit_result'
stackplot(
  x,
  xlim = NULL,
  y_offset = 0,
  dyn = 1,
  x_pos = 1,
  y_pos = 1,
  z_pos = 1,
  coil = 1,
  n = NULL,
  sub_bl = FALSE,
  labels = FALSE,
  label_names = NULL,
  sig_col = "black",
  restore_def_par = TRUE,
  omit_signals = NULL,
  combine_lipmm = FALSE,
  combine_metab = FALSE,
  mar = NULL,
  ...
)
```

Arguments

`x`	fit_result object.
`xlim`	the range of values to display on the x-axis, eg xlim = c(4,1).
`y_offset`	separate basis signals in the y-axis direction by this value.
`dyn`	the dynamic index to plot.
`x_pos`	the x index to plot.
`y_pos`	the y index to plot.
`z_pos`	the z index to plot.
`coil`	the coil element number to plot.
`n`	single index element to plot (overrides other indices when given).

sub_bl	subtract the baseline from the data and fit (logical).
labels	print signal labels at the right side of the plot.
label_names	provide a character vector of signal names to replace the defaults determined from the basis set.
sig_col	colour of individual signal components.
restore_def_par	restore default plotting par values after the plot has been made.
omit_signals	a character vector of basis signal names to be removed from the plot.
combine_lipmm	combine all basis signals with names starting with "Lip" or "MM".
combine_metab	combine all basis signals with names not starting with "Lip" or "MM".
mar	option to adjust the plot margins. See ?par.
...	further arguments to plot method.

`stackplot.mrs_data` *Stackplot plotting method for objects of class mrs_data.*

Description

Stackplot plotting method for objects of class mrs_data.

Usage

```
## S3 method for class 'mrs_data'
stackplot(
  x,
  xlim = NULL,
  mode = "re",
  x_units = NULL,
  fd = TRUE,
  col = NULL,
  x_offset = 0,
  y_offset = 0,
  dim = "dyn",
  x_pos = NULL,
  y_pos = NULL,
  z_pos = NULL,
  dyn = 1,
  coil = 1,
  bty = NULL,
  labels = NULL,
  lab_cex = 1,
  right_marg = NULL,
  bl_lty = NULL,
  restore_def_par = TRUE,
  ...
)
```

Arguments

x	object of class mrs_data.
xlim	the range of values to display on the x-axis, eg xlim = c(4,1).
mode	representation of the complex numbers to be plotted, can be one of: "re", "im", "mod" or "arg".
x_units	the units to use for the x-axis, can be one of: "ppm", "hz", "points" or "seconds".
fd	display data in the frequency-domain (default), or time-domain (logical).
col	set the colour of the line, eg col = rgb(1,0,0,0.5).
x_offset	separate plots in the x-axis direction by this value. Default value is 0.
y_offset	separate plots in the y-axis direction by this value.
dim	the dimension to stack in the y-axis direction, can be one of: "dyn", "x", "y", "z" or "coil".
x_pos	the x index to plot.
y_pos	the y index to plot.
z_pos	the z index to plot.
dyn	the dynamic index to plot.
coil	the coil element number to plot.
bty	option to draw a box around the plot. See ?par.
labels	add labels to each data item.
lab_cex	label size.
right_marg	change the size of the right plot margin.
bl_lty	linetype for the y = 0 baseline trace. A default value NULL results in no baseline being plotted.
restore_def_par	restore default plotting par values after the plot has been made.
...	other arguments to pass to the matplot method.

---

sum_coils *Calculate the sum across receiver coil elements.*

Description

Calculate the sum across receiver coil elements.

Usage

```
sum_coils(mrs_data)
```

Arguments

mrs_data	MRS data split across receiver coil elements.

Value

sum across coil elements.

sum_dyns *Calculate the sum of data dynamics.*

Description

Calculate the sum of data dynamics.

Usage

```
sum_dyns(mrs_data)
```

Arguments

mrs_data	dynamic MRS data.

Value

sum of data dynamics.

td2fd *Transform time-domain data to the frequency-domain.*

Description

Transform time-domain data to the frequency-domain.

Usage

```
td2fd(mrs_data)
```

Arguments

mrs_data	MRS data in time-domain representation.

Value

MRS data in frequency-domain representation.

tdsr *Time-domain spectral registration.*

Description

An implementation of the method published by Near et al MRM 73:44-50 (2015).

Usage

```
tdsr(mrs_data, ref = NULL, xlim = c(4, 0.5), max_t = 0.2)
```

Arguments

mrs_data	MRS data to be corrected.
ref	optional MRS data to use as a reference, the mean of all dynamics is used if this argument is not supplied.
xlim	optional frequency range to perform optimisation, set to NULL to use the full range.
max_t	truncate the FID when longer than max_t to reduce time taken.

Value

a list containing the corrected data; phase and shift values in units of degrees and Hz respectively.

td_conv_filt *Time-domain convolution based filter.*

Description

Time-domain convolution based filter described by: Marion D, Ikura M, Bax A. Improved solvent suppression in one-dimensional and twodimensional NMR spectra by convolution of time-domain data. J Magn Reson 1989;84:425-430.

Usage

```
td_conv_filt(mrs_data, K = 25, ext = 1)
```

Arguments

mrs_data	MRS data to be filtered.
K	window width in data points.
ext	point separation for linear extrapolation.

varpro_3_para_opts *Return a list of options for VARPRO based fitting with 3 free parameters:*

- *zero'th order phase correction*
- *global damping*
- *global frequency shift.*

Description

Return a list of options for VARPRO based fitting with 3 free parameters:

- zero'th order phase correction
- global damping
- global frequency shift.

Usage

```
varpro_3_para_opts(
  nstart = 20,
  init_damping = 2,
  maxiters = 200,
  max_shift = 5,
  max_damping = 5,
  anal_jac = FALSE,
  bl_smth_pts = 80
)
```

Arguments

nstart	position in the time-domain to start fitting, units of data points.
init_damping	starting value for the global Gaussian line-broadening term - measured in Hz.
maxiters	maximum number of levmar iterations to perform.
max_shift	maximum global shift allowed, measured in Hz.
max_damping	maximum damping allowed, FWHM measured in Hz.
anal_jac	option to use the analytic or numerical Jacobian (logical).
bl_smth_pts	number of data points to use in the baseline smoothing calculation.

Value

list of options.

varpro_opts *Return a list of options for VARPRO based fitting.*

Description

Return a list of options for VARPRO based fitting.

Usage

```
varpro_opts(
  nstart = 20,
  init_g_damping = 2,
  maxiters = 200,
  max_shift = 5,
  max_g_damping = 5,
  max_ind_damping = 5,
  anal_jac = TRUE,
  bl_smth_pts = 80
)
```

Arguments

nstart	position in the time-domain to start fitting, units of data points.
init_g_damping	starting value for the global Gaussian line-broadening term - measured in Hz.
maxiters	maximum number of levmar iterations to perform.
max_shift	maximum shift allowed to each element in the basis set, measured in Hz.
max_g_damping	maximum permitted global Gaussian line-broadening.
max_ind_damping	maximum permitted Lorentzian line-broadening for each element in the basis set, measured in Hz.
anal_jac	option to use the analytic or numerical Jacobian (logical).
bl_smth_pts	number of data points to use in the baseline smoothing calculation.

Value

list of options.

Examples

```
varpro_opts(nstart = 10)
```

vec2mrs_data *Convert a vector into a mrs_data object.*

Description

Convert a vector into a mrs_data object.

Usage

```
vec2mrs_data(
  vec,
  fs = def_fs(),
  ft = def_ft(),
  ref = def_ref(),
  dyns = 1,
  fd = FALSE
)
```

Arguments

vec	the data vector.
fs	sampling frequency in Hz.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.
dyns	replicate the data across the dynamic dimension.
fd	flag to indicate if the matrix is in the frequency domain (logical).

Value

mrs_data object.

write_basis *Write a basis object to an LCModel .basis formatted file.*

Description

Write a basis object to an LCModel .basis formatted file.

Usage

```
write_basis(basis, basis_file, fwhmba = 0.1)
```

Arguments

basis	basis object to be exported.
basis_file	path to basis file to be generated.
fwhmba	parameter used by LCModel.

---

write_basis_tqn *Generate a basis file using TARQUIN.*

Description

Generate a basis file using TARQUIN.

Usage

```
write_basis_tqn(basis_file, metab_data, opts = NULL)
```

Arguments

basis_file	filename of the basis file to be generated.
metab_data	MRS data object to match the generated basis parameters.
opts	list of options to pass to TARQUIN.

Examples

```
## Not run:
write_basis_tqn('test.basis',mrs_data,c("--echo","0.04"))

## End(Not run)
```

---

write_mrs *Write MRS data object to file.*

Description

Write MRS data object to file.

Usage

```
write_mrs(fname, mrs_data, format = NULL)
```

Arguments

fname	the filename of the output.
mrs_data	object to be written to file.
format	string describing the data format. Must be one of the following : "nifti", "dpt", "lcm_raw", "rds". If not specified, the format will be guessed from the filename extension.

write_mrs_nifti *Write MRS data object to file in NIFTI format.*

Description

Write MRS data object to file in NIFTI format.

Usage

```
write_mrs_nifti(fname, mrs_data)
```

Arguments

fname	the filename of the output NIFTI MRS data.
mrs_data	object to be written to file.

zero_nzoc *Zero all non-zero-order coherences.*

Description

Zero all non-zero-order coherences.

Usage

```
zero_nzoc(sys, rho)
```

Arguments

sys	spin system object.
rho	density matrix.

Value

density matrix.

zf — *Zero-fill MRS data in the time domain.*

Description

Zero-fill MRS data in the time domain.

Usage

```
zf(x, factor = 2)

## S3 method for class 'mrs_data'
zf(x, factor = 2)

## S3 method for class 'basis_set'
zf(x, factor = 2)
```

Arguments

x	input mrs_data or basis_set object.
factor	zero-filling factor, factor of 2 returns a dataset with twice the original data points.

Value

zero-filled data.

zf_xy — *Zero-fill MRSI data in the k-space x-y direction.*

Description

Zero-fill MRSI data in the k-space x-y direction.

Usage

```
zf_xy(mrs_data, factor = 2)
```

Arguments

mrs_data	MRSI data.
factor	zero-filling factor, factor of 2 returns a dataset with twice the original points in the x-y directions.

Value

zero-filled data.

Index