Package ‘spant’

May 30, 2024

Type Package

Title MR Spectroscopy Analysis Tools

Version 2.21.0

Date 2024-05-30

Description Tools for reading, visualising and processing Magnetic Resonance Spectroscopy data. The package includes methods for spectral fitting: Wilson (2021) <DOI:10.1002/mrm.28385> and spectral alignment: Wilson (2018) <DOI:10.1002/mrm.27605>.

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

License GPL-3

RoxygenNote 7.2.3

NeedsCompilation yes

LazyData yes

Depends R (>= 2.10)

Imports abind, plyr, pracma, stringr, expm, signal, minpack.lm, utils, graphics, grDevices, ptw, mmand, RNifti, RNiftyReg, fields, numDeriv, nloptr, irlba, jsonlite

Suggests viridisLite, shiny, ggplot2, miniUI, knitr, rmarkdown, testthat, ragg, doParallel

VignetteBuilder knitr

Encoding UTF-8

Language en-GB

URL `https://martin3141.github.io/spant/`,
`https://github.com/martin3141/spant/`

Author Martin Wilson [cre, aut] (<`https://orcid.org/0000-0002-2089-3956`>),
Yong Wang [ctb],
John Muschelli [ctb]

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

Repository CRAN

Date/Publication 2024-05-30 07:00:08 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`> (ORCID)

Other contributors:

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

See Also

Useful links:

- https://martin3141.github.io/spant/
- https://github.com/martin3141/spant/
- 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 = 0.078,
  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 = 0.0078,
  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 = 0.0078,
  max_basis_damping_broad = 2,
  ahat_calc_method = "lh_pnnls",
  prefit_phase_search = TRUE,
  freq_reg = NULL,
  lb_reg = NULL,
  output_all_paras = FALSE,
  output_all_paras_raw = FALSE,
  input_paras_raw = NULL,
  optim_lw_only = FALSE,
  optim_lw_only_limit = 20,
  lb_init = 0.001
)
```

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 (ppm).
`max_damping`	maximum permitted value of the global damping parameter (Hz).
`max_phase`	the maximum absolute permitted value of the global zero-order phase term (degrees). Note, the prefit_phase_search option is not constrained by this term.
`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 (ppm).
`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 (ppm). 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".
`prefit_phase_search`	perform a 1D search for the optimal phase in the prefit stage of the algorithm.
`freq_reg`	frequency shift parameter.
`lb_reg`	individual line broadening parameter.

`output_all_paras`	include more fitting parameters in the fit table, e.g. individual shift and damping factors for each basis set element.
`output_all_paras_raw`	include raw fitting parameters in the fit table. For advanced diagnostic use only.
`input_paras_raw`	input raw fitting parameters. For advanced diagnostic use only.
`optim_lw_only`	optimize the global line-broadening term only.
`optim_lw_only_limit`	limits for the line-breading term as a percentage of the starting value when optim_lw_only is TRUE.
`lb_init`	initial Lorentzian line broadening value for the individual basis signals. Setting to 0 will clash with the minimum allowable value (eg hard constraint) during the detailed fit.

Value

full list of options.

Examples

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

abfit_opts_v1_9_0 *Return a list of options for an ABfit analysis to maintain comparability with analyses performed with version 1.9.0 (and earlier) of spant.*

Description

Return a list of options for an ABfit analysis to maintain comparability with analyses performed with version 1.9.0 (and earlier) of spant.

Usage

```
abfit_opts_v1_9_0(...)
```

Arguments

`...`	arguments passed to abfit_opts.

Value

full list of options.

acquire *Simulate pulse sequence acquisition.*

Description

Simulate pulse sequence acquisition.

Usage

```
acquire(sys, rec_phase = 0, tol = 1e-04, detect = NULL, amp_scale = 1)
```

Arguments

`sys`	spin system object.
`rec_phase`	receiver phase in degrees.
`tol`	ignore resonance amplitudes below this threshold.
`detect`	detection nuclei.
`amp_scale`	scaling factor for the output amplitudes.

Value

a list of resonance amplitudes and frequencies.

add_noise *Add noise to an mrs_data object.*

Description

Add noise to an mrs_data object.

Usage

```
add_noise(mrs_data, sd = 0.1, fd = TRUE)
```

Arguments

`mrs_data`	data to add noise to.
`sd`	standard deviation of the noise.
`fd`	generate the noise samples in the frequency-domain (TRUE) or time-domain (FALSE). This is required since the absolute value of the standard deviation of noise samples changes when data is Fourier transformed.

Value

mrs_data object with additive normally distributed noise.

`add_noise_spec_snr` *Add noise to an mrs_data object to match a given SNR.*

Description

Add noise to an mrs_data object to match a given SNR.

Usage

```
add_noise_spec_snr(mrs_data, target_snr, sig_region = c(4, 0.5))
```

Arguments

`mrs_data`	data to add noise to.
`target_snr`	desired spectral SNR, note this assumes the input data is noise-free, eg simulated data. Note the SNR is estimated from the first scan in the dataset and the same noise level is added to all spectra.
`sig_region`	spectral limits to search for the strongest spectral data point.

Value

mrs_data object with additive normally distributed noise.

`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,
  ref_amp = 1,
  zf_factor = 2,
  lb = 2,
  max_shift = 20,
  ret_df = FALSE,
  mean_dyns = FALSE
)
```

Arguments

`mrs_data`	data to be aligned.
`ref_freq`	reference frequency in ppm units. More than one frequency may be specified.
`ref_amp`	amplitude value for the reference signal. More than one value may be specified to match the number of ref_freq signals.
`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).
`mean_dyns`	align the mean spectrum and apply the same shift to each dynamic.

Value

aligned data object.

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

Description

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

Usage

```
apodise_xy(mrs_data, func = "hamming", w = 2.5)
```

Arguments

`mrs_data`	MRSI data.
`func`	must be "hamming" or "gaussian".
`w`	the reciprocal of the standard deviation for the Gaussian function.

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.

`append_regs` *Append multiple regressor data frames into a single data frame.*

Description

Append multiple regressor data frames into a single data frame.

Usage

```
append_regs(...)
```

Arguments

`...`	input regressor data frames.

Value

output regressor data frame.

`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_pulse` *Simulate an RF pulse on a single spin.*

Description

Simulate an RF pulse on a single spin.

Usage

```
apply_pulse(sys, rho, spin_n, angle, nuc, xy)
```

Arguments

`sys`	spin system object.
`rho`	density matrix.
`spin_n`	spin index.
`angle`	RF flip angle in degrees.
`nuc`	nucleus influenced by the pulse.
`xy`	x or y pulse.

Value

density matrix.

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(),
  nuc = def_nuc(),
  fd = FALSE
)
```

Arguments

`data_array`	7d data array.
`fs`	sampling frequency in Hz.
`ft`	transmitter frequency in Hz.
`ref`	reference value for ppm scale.
`nuc`	nucleus that is resonant at the transmitter frequency.
`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 = c(4, 1.8), smo_ppm_sd = 0.05, ret_phase = FALSE)
```

Arguments

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

Value

MRS data object and phase values (optional).

back_extrap_ar *Back extrapolate time-domain data points using an autoregressive model.*

Description

Back extrapolate time-domain data points using an autoregressive model.

Usage

```
back_extrap_ar(
  mrs_data,
  extrap_pts,
  pred_pts = NULL,
  method = "burg",
  rem_add = TRUE,
  ...
)
```

Arguments

`mrs_data`	mrs_data object.
`extrap_pts`	number of points to extrapolate.
`pred_pts`	number of points to base the extrapolation on.
`method`	character string specifying the method to fit the model. Must be one of the strings in the default argument (the first few characters are sufficient). Defaults to "burg".
`rem_add`	remove additional points from the end of the FID to maintain the original length of the dataset. Default to TRUE.
`...`	additional arguments to specific methods, see ?ar.

Value

back extrapolated data.

basis2dyn_mrs_data *Convert a basis object to a dynamic mrs_data object.*

Description

Convert a basis object to a dynamic mrs_data object.

Usage

```
basis2dyn_mrs_data(basis, amps, tr)
```

Arguments

`basis`	basis set object.
`amps`	a data frame with each column corresponding to a basis element and each row corresponding to each dynamic scan.
`tr`	the dataset repetition time in seconds.

Value

a dynamic mrs_data object.

`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, amps = NULL, shifts = NULL)
```

Arguments

`basis`	basis set object.
`sum_elements`	return the sum of basis elements (logical)
`amps`	a vector of scaling factors to apply to each basis element.
`shifts`	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

Eilers P. H. C. and Boelens H. F. M. (2005) Baseline correction with asymmetric least squares smoothing. Leiden Univ. Medical Centre Report.

Usage

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

Arguments

`mrs_data`	mrs_data object.
`lambda`	controls the baseline flexibility.
`p`	controls the penalty for negative data points.
`ret_bc_only`	return the baseline corrected data only. When FALSE the baseline estimate and input data will be returned.

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.

bc_gauss

Apply and subtract a Gaussian smoother in the spectral domain.

Description

Apply and subtract a Gaussian smoother in the spectral domain.

Usage

```
bc_gauss(mrs_data, smo_ppm_sd)
```

Arguments

`mrs_data`	mrs_data object.
`smo_ppm_sd`	Gaussian smoother sd in ppm units.

Value

smoother subtracted data.

`bc_poly` *Fit and subtract a polynomial to each spectrum in a dataset.*

Description

Fit and subtract a polynomial to each spectrum in a dataset.

Usage

```
bc_poly(mrs_data, p_deg = 1)
```

Arguments

`mrs_data`	mrs_data object.
`p_deg`	polynomial degree.

Value

polynomial subtracted data.

`bc_spline` *Fit and subtract a smoothing spline to each spectrum in a dataset.*

Description

Fit and subtract a smoothing spline to each spectrum in a dataset.

Usage

```
bc_spline(mrs_data, spar = 0.5, nknots = 100)
```

Arguments

`mrs_data`	mrs_data object.
`spar`	smoothing parameter typically between 0 and 1.
`nknots`	number of spline knots.

Value

smoothing spline subtracted 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.

bin_spec *Bin equally spaced spectral regions.*

Description

Bin equally spaced spectral regions.

Usage

```
bin_spec(mrs_data, width = 0.05, unit = "ppm")
```

Arguments

`mrs_data`	data to be "binned".
`width`	bin width.
`unit`	bin width unit, can be "ppm" (default) or "pts".

Value

binned mrs_data object.

`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_design_efficiency` *Calculate the efficiency of a regressor data frame.*

Description

Calculate the efficiency of a regressor data frame.

Usage

```
calc_design_efficiency(regressor_df, contrasts)
```

Arguments

`regressor_df`	input regressor data frame.
`contrasts`	a vector of contrast values.

`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()
```

circ_mask — *Create a logical circular mask spanning the full extent of an n x n matrix.*

Description

Create a logical circular mask spanning the full extent of an n x n matrix.

Usage

```
circ_mask(d, n, offset = 1)
```

Arguments

`d`	diameter of the mask.
`n`	number of matrix rows and columns.
`offset`	offset the mask centre in matrix dimension units.

Value

logical n x n mask matrix.

coherence_filter — *Zero all coherence orders other than the one supplied as an argument.*

Description

Zero all coherence orders other than the one supplied as an argument.

Usage

```
coherence_filter(sys, rho, order = 0)
```

Arguments

`sys`	spin system object.
`rho`	density matrix.
`order`	coherence order to keep (default is 0).

Value

density matrix.

`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, rm_masked = FALSE)

## S3 method for class 'mrs_data'
collapse_to_dyns(x, rm_masked = FALSE)

## S3 method for class 'fit_result'
collapse_to_dyns(x, rm_masked = FALSE)
```

Arguments

`x`	data object to be collapsed (mrs_data or fit_result object).
`rm_masked`	remove masked dynamics from the output.

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,
  scale_method = "sig_noise_sq",
  sum_coils = TRUE,
  noise_region = c(-0.5, -2.5),
  average_ref_dyns = TRUE,
  ref_pt_index = 1,
  ret_metab_only = FALSE
)
```

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).
`scale_method`	one of "sig_noise_sq", "sig_noise" or "sig".
`sum_coils`	sum the coil elements as a final step (logical).
`noise_region`	the spectral region (in ppm) to estimate the noise.
`average_ref_dyns`	take the mean of the reference scans in the dynamic dimension before use.
`ref_pt_index`	time-domain point to use for estimating phase and scaling values.
`ret_metab_only`	return the metabolite data only, even if reference data has been specified.

Value

MRS data.

`comb_fit_list_fit_tables` *Combine all fitting data points from a list of fits into a single data frame.*

Description

Combine all fitting data points from a list of fits into a single data frame.

Usage

```
comb_fit_list_fit_tables(
  fit_list,
  add_extra = TRUE,
  harmonise_ppm = TRUE,
  inc_basis_sigs = FALSE,
  inc_indices = TRUE,
  add_res_id = TRUE
)
```

Arguments

`fit_list`	list of fit_result objects.
`add_extra`	add variables in the extra data frame to the output (TRUE).
`harmonise_ppm`	ensure the ppm scale for each fit is identical to the first.
`inc_basis_sigs`	include the individual fitting basis signals in the output table, defaults to FALSE.
`inc_indices`	include indices such as X, Y and coil in the output, defaults to TRUE. These are generally not useful for SVS analysis.
`add_res_id`	add a res_id column to the output to distinguish between datasets.

Value

a data frame containing the fit data points.

comb_fit_list_result_tables

Combine the fit result tables from a list of fit results.

Description

Combine the fit result tables from a list of fit results.

Usage

```
comb_fit_list_result_tables(fit_list, add_extra = TRUE, add_res_id = TRUE)
```

Arguments

`fit_list`	a list of fit_result objects.
`add_extra`	add variables in the extra data frame to the output (TRUE).
`add_res_id`	add a res_id column to the output to distinguish between datasets.

Value

a data frame combine all fit result tables with an additional id column to differentiate between data sets. Any variables in the extra data frame may be optionally added to the result.

comb_fit_tables *Combine all fitting data points into a single data frame.*

Description

Combine all fitting data points into a single data frame.

Usage

```
comb_fit_tables(fit_res, inc_basis_sigs = FALSE, inc_indices = TRUE)
```

Arguments

fit_res	a single fit_result object.
inc_basis_sigs	include the individual fitting basis signals in the output table, defaults to FALSE.
inc_indices	include indices such as X, Y and coil in the output, defaults to TRUE. These are generally not useful for SVS analysis.

Value

a data frame containing the fit data points.

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_basis` *Crop* `basis_set` *object based on a frequency range.*

Description

Crop `basis_set` object based on a frequency range.

Usage

```
crop_basis(basis, xlim = c(4, 0.2), scale = "ppm")
```

Arguments

`basis`	basis_set object to be cropped in the spectral dimension.
`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_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_td_pts_end — *Crop `mrs_data` object data points at the end of the FID.*

Description

Crop `mrs_data` object data points at the end of the FID.

Usage

```
crop_td_pts_end(mrs_data, pts)
```

Arguments

`mrs_data`	MRS data.
`pts`	number of points to remove from the end of the FID.

Value

cropped `mrs_data` object.

crop_td_pts_pot — *Crop `mrs_data` object data points in the time-domain rounding down to the next smallest power of two (pot). Data that already has a pot length will not be changed.*

Description

Crop `mrs_data` object data points in the time-domain rounding down to the next smallest power of two (pot). Data that already has a pot length will not be changed.

Usage

```
crop_td_pts_pot(mrs_data)
```

Arguments

`mrs_data`	MRS data.

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_fd *Decimate an MRS signal to half the original sampling frequency by filtering in the frequency domain before down sampling.*

Description

Decimate an MRS signal to half the original sampling frequency by filtering in the frequency domain before down sampling.

Usage

```
decimate_mrs_fd(mrs_data)
```

Arguments

`mrs_data`	MRS data object.

Value

decimated data at half the original sampling frequency.

decimate_mrs_td *Decimate an MRS signal by filtering in the time domain before downsampling.*

Description

Decimate an MRS signal by filtering in the time domain before downsampling.

Usage

```
decimate_mrs_td(mrs_data, q = 2, n = 4, ftype = "iir")
```

Arguments

mrs_data	MRS data object.
q	integer factor to downsample by (default = 2).
n	filter order used in the downsampling.
ftype	filter type, "iir" or "fir".

Value

decimated data.

deconv_mrs *Deconvolve two MRS data objects.*

Description

Deconvolve two MRS data objects.

Usage

```
deconv_mrs(mrs_data_a, mrs_data_b)
```

Arguments

mrs_data_a	MRS data to be deconvolved.
mrs_data_b	MRS data to be deconvolved.

Value

deconvolved 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"),
  nuc = getOption("spant.def_nuc")
)
```

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.
`nuc`	specify the resonant nucleus.

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.
- nuc resonant nucleus.

`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.

`dicom_reader` *A very simple DICOM reader.*

Description

Note this reader is very basic and does not use a DICOM dictionary or try to convert the data to the correct datatype. For a more robust and sophisticated reader use the oro.dicom package.

Usage

```
dicom_reader(
  input,
  tags = list(sop_class_uid = "0008,0016"),
  endian = "little",
  debug = FALSE
)
```

Arguments

`input`	either a file path or raw binary object.
`tags`	a named list of tags to be extracted from the file. eg tags <- list(spec_data = "7FE1,1010", pat_name = "0010,0010")
`endian`	can be "little" or "big".
`debug`	print out some debugging information, can be "little" or "big".

Value

a list with the same structure as the input, but with tag codes replaced with the corresponding data in a raw format.

`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_fd — *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_fd(mrs_data)
```

Arguments

`mrs_data`	MRS data object.

Value

downsampled data.

downsample_mrs_td — *Downsample an MRS signal by a factor of 2 by removing every other data point in the time-domain. Note, signals outside the new sampling frequency will be aliased.*

Description

Downsample an MRS signal by a factor of 2 by removing every other data point in the time-domain. Note, signals outside the new sampling frequency will be aliased.

Usage

```
downsample_mrs_td(mrs_data)
```

Arguments

`mrs_data`	MRS data object.

Value

downsampled data.

`dyn_acq_times` *Return a time scale vector of acquisition times for a dynamic MRS scan. The first temporal scan is assigned a value of 0.*

Description

Return a time scale vector of acquisition times for a dynamic MRS scan. The first temporal scan is assigned a value of 0.

Usage

```
dyn_acq_times(mrs_data)
```

Arguments

`mrs_data`	MRS data.

Value

time scale vector in units of seconds.

`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.

elliptical_mask — *Create an elliptical mask stored as a matrix of logical values.*

Description

Create an elliptical mask stored as a matrix of logical values.

Usage

```
ellipctical_mask(xN, yN, x0, y0, xr, yr, angle)
```

Arguments

`xN`	number of pixels in the x dimension.
`yN`	number of pixels in the y dimension.
`x0`	centre of ellipse in the x direction in units of pixels.
`y0`	centre of ellipse in the y direction in units of pixels.
`xr`	radius in the x direction in units of pixels.
`yr`	radius in the y direction in units of pixels.
`angle`	angle of rotation in degrees.

Value

logical mask matrix with dimensions fov_yN x fov_xN.

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.

`fd_gauss_smo` *Apply a Gaussian smoother in the spectral domain.*

Description

Apply a Gaussian smoother in the spectral domain.

Usage

```
fd_gauss_smo(mrs_data, smo_ppm_sd)
```

Arguments

`mrs_data`	mrs_data object.
`smo_ppm_sd`	Gaussian smoother sd in ppm units.

Value

spectrally smoothed data.

`find_bids_mrs` *Search for MRS data files in a BIDS filesystem structure.*

Description

Search for MRS data files in a BIDS filesystem structure.

Usage

```
find_bids_mrs(path)
```

Arguments

`path`	path to the directory containing the BIDS structure.

Value

data frame containing full paths and information on each MRS file.

find_mrs_files *Find valid MRS data files recursively from a directory path.*

Description

Find valid MRS data files recursively from a directory path.

Usage

```
find_mrs_files(dir)
```

Arguments

dir	a directory path.

Value

a vector of valid MRS data files.

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",
  extra = NULL
)
```

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.
`extra`	an optional data frame to provide additional variables for use in subsequent analysis steps, eg id or grouping variables.

Details

Fitting approaches described in the following references: ABfit Wilson, M. Adaptive baseline fitting for 1H MR spectroscopy analysis. Magn Reson Med 2012;85:13-29.

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)
## 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).

`fit_t1_ti_array` *Fit a T1 recovery curve, from multiple TIs, to a set of amplitudes.*

Description

Fit a T1 recovery curve, from multiple TIs, to a set of amplitudes.

Usage

```
fit_t1_ti_array(
  ti_vec,
  amp_vec,
  lower = 0,
  upper = 10,
  output_fit_res = 0.01,
  ret_full = TRUE
)
```

Arguments

`ti_vec`	vector of TI values in seconds.
`amp_vec`	vector of amplitudes.
`lower`	minimum allowable T1 value.
`upper`	maximum allowable T1 value.
`output_fit_res`	temporal resolution (seconds) of the ideal output relaxation curve.
`ret_full`	return full fitting information including ideal relaxation curve.

Value

a list containing relaxation parameters and an ideal curve for fit evaluation.

`fit_t1_tr_array` *Fit a T1 recovery curve, from multiple TRs, to a set of amplitudes.*

Description

Fit a T1 recovery curve, from multiple TRs, to a set of amplitudes.

Usage

```
fit_t1_tr_array(
  tr_vec,
  amp_vec,
  lower = 0,
  upper = 10,
  output_fit_res = 0.01,
  ret_full = TRUE
)
```

Arguments

`tr_vec`	vector of TR values in seconds.
`amp_vec`	vector of amplitudes.
`lower`	minimum allowable T1 value.
`upper`	maximum allowable T1 value.
`output_fit_res`	temporal resolution (seconds) of the ideal output relaxation curve.
`ret_full`	return full fitting information including ideal relaxation curve.

Value

a list containing relaxation parameters and an ideal curve for fit evaluation.

fit_t2_te_array — *Fit a T2 relaxation curve, from multiple TEs, to a set of amplitudes.*

Description

Fit a T2 relaxation curve, from multiple TEs, to a set of amplitudes.

Usage

```
fit_t2_te_array(
  te_vec,
  amp_vec,
  lower = 0,
  upper = 10,
  output_fit_res = 0.01,
  ret_full = TRUE
)
```

Arguments

`te_vec`	vector of TE values in seconds.
`amp_vec`	vector of amplitudes.
`lower`	minimum allowable T2 value.
`upper`	maximum allowable T2 value.
`output_fit_res`	temporal resolution (seconds) of the ideal output relaxation curve.
`ret_full`	return full fitting information including ideal relaxation curve.

Value

a list containing relaxation parameters and an ideal curve for fit evaluation.

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.

`fp_scale` *Scale the first time-domain data point in an mrs_data object.*

Description

Scale the first time-domain data point in an mrs_data object.

Usage

```
fp_scale(mrs_data, scale = 0.5)
```

Arguments

`mrs_data`	MRS data.
`scale`	scaling value, defaults to 0.5.

Value

scaled mrs_data object.

`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_dyns` *Apply the Fourier transform over the dynamic dimension.*

Description

Apply the Fourier transform over the dynamic dimension.

Usage

```
ft_dyns(mrs_data, ft_shift = FALSE, ret_mod = FALSE, fd = TRUE)
```

Arguments

`mrs_data`	MRS data where the dynamic dimension is in the time-domain.
`ft_shift`	apply FT shift to the output, default is FALSE.
`ret_mod`	return the modulus out the transform, default is FALSE.
`fd`	transform the chemical shift axis to the frequency domain first, default is TRUE.

Value

transformed MRS data.

`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.

gausswin_2d — *Create a two dimensional Gaussian window function stored as a matrix.*

Description

Create a two dimensional Gaussian window function stored as a matrix.

Usage

```
gausswin_2d(xN, yN, x0, y0, xw, yw)
```

Arguments

`xN`	number of pixels in the x dimension.
`yN`	number of pixels in the y dimension.
`x0`	centre of window function in the x direction in units of pixels. Note, only integer values are applied.
`y0`	centre of window function in the y direction in units of pixels. Note, only integer values are applied.
`xw`	the reciprocal of the standard deviation of the Gaussian window in x direction.
`yw`	the reciprocal of the standard deviation of the Gaussian window in y direction.

Value

matrix with dimensions fov_yN x fov_xN.

gen_baseline_reg — *Generate baseline regressor.*

Description

Generate baseline regressor.

Usage

```
gen_baseline_reg(mrs_data)
```

Arguments

`mrs_data`	mrs_data object for timing information.

Value

a single baseline regressor with value of 1.

gen_bold_reg *Generate BOLD regressors.*

Description

Generate BOLD regressors.

Usage

```
gen_bold_reg(
  onset,
  duration = NULL,
  trial_type = NULL,
  mrs_data = NULL,
  match_tr = TRUE,
  dt = 0.1,
  normalise = FALSE
)
```

Arguments

`onset`	stimulus onset in seconds.
`duration`	stimulus duration in seconds.
`trial_type`	string label for the stimulus.
`mrs_data`	mrs_data object for timing information.
`match_tr`	match the output to the input mrs_data.
`dt`	timing resolution for internal calculations.
`normalise`	normalise the response function to have a maximum value of one.

Value

BOLD regressor data frame.

gen_conv_reg *Generate regressors by convolving a specified response function with a stimulus.*

Description

Generate regressors by convolving a specified response function with a stimulus.

Usage

```
gen_conv_reg(
  onset,
  duration = NULL,
  trial_type = NULL,
  mrs_data = NULL,
  resp_fn,
  match_tr = TRUE,
  normalise = FALSE
)
```

Arguments

`onset`	stimulus onset in seconds.
`duration`	stimulus duration in seconds.
`trial_type`	string label for the stimulus.
`mrs_data`	mrs_data object for timing information.
`resp_fn`	a data frame specifying the response function to be convolved.
`match_tr`	match the output to the input mrs_data.
`normalise`	normalise the response function to have a maximum value of one.

Value

BOLD regressor data frame.

`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.

gen_I *Generate the I product operator for a single spin.*

Description

Generate the I product operator for a single spin.

Usage

```
gen_I(n, spin_num, op)
```

Arguments

`n`	spin index number for the required operator.
`spin_num`	vector of spin numbers in the system.
`op`	operator, one of "x", "y", "z", "p", "m".

Value

I product operator matrix.

`gen_impulse_reg` *Generate impulse regressors.*

Description

Generate impulse regressors.

Usage

```
gen_impulse_reg(onset, trial_type = NULL, mrs_data = NULL)
```

Arguments

`onset`	stimulus onset in seconds.
`trial_type`	string label for the stimulus.
`mrs_data`	mrs_data object for timing information.

Value

impulse regressors data frame.

`gen_poly_reg` *Generate polynomial regressors.*

Description

Generate polynomial regressors.

Usage

```
gen_poly_reg(mrs_data, degree)
```

Arguments

`mrs_data`	mrs_data object for timing information.
`degree`	the degree of the polynomial.

Value

polynomial regressors.

gen_trap_reg *Generate trapezoidal regressors.*

Description

Generate trapezoidal regressors.

Usage

```
gen_trap_reg(
  onset,
  duration,
  trial_type = NULL,
  mrs_data = NULL,
  rise_t = 0,
  fall_t = 0,
  exp_fall = FALSE,
  exp_fall_power = 1,
  smo_sigma = NULL,
  match_tr = TRUE,
  dt = 0.01,
  normalise = FALSE
)
```

Arguments

`onset`	stimulus onset in seconds.
`duration`	stimulus duration in seconds.
`trial_type`	string label for the stimulus.
`mrs_data`	mrs_data object for timing information.
`rise_t`	time to reach a plateau from baseline in seconds.
`fall_t`	time to fall from plateau level back to baseline in seconds.
`exp_fall`	model an exponential fall instead of linear.
`exp_fall_power`	exponential fall power.
`smo_sigma`	standard deviation of Gaussian smoothing kernel in seconds. Set to NULL to disable (default behavior).
`match_tr`	match the output to the input mrs_data.
`dt`	timing resolution for internal calculations.
`normalise`	normalise the response function to have a maximum value of one.

Value

trapezoidal regressor data frame.

get_1h_braino_basis_names
Return a character vector of molecules included in the GE BRAINO phantom.

Description

Return a character vector of molecules included in the GE BRAINO phantom.

Usage

```
get_1h_braino_basis_names()
```

Value

a character vector of molecule names.

get_1h_brain_basis_names
Return a character vector of common 1H molecules found in healthy human brain.

Description

Note, this is a basic set and it may be appropriate to also include Asc, Gly and PEth for high quality MRS data.

Usage

```
get_1h_brain_basis_names(add = NULL, remove = NULL, inc_lip_mm = TRUE)
```

Arguments

`add`	optional character vector of additional molecular names. Eg c("asc", "gly", "peth").
`remove`	optional character vector of molecular names to remove from the set. Eg c("m_cr_ch2").
`inc_lip_mm`	include Lipid and MM basis signals.

Value

a character vector of molecule names.

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_1h_spectre_basis_names — *Return a character vector of molecules included in the Gold Star Phantoms SPECTRE phantom.*

Description

Return a character vector of molecules included in the Gold Star Phantoms SPECTRE phantom.

Usage

```
get_1h_spectre_basis_names()
```

Value

a character vector of molecule names.

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,
  ensure_odd = TRUE
)
```

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).
`ensure_odd`	add 1mm to the FOV when required to ensure the output pdf has odd dimensions. Required when using get_mrsi2d_seg.

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_basis_subset` *Return a subset of the input basis.*

Description

Return a subset of the input basis.

Usage

```
get_basis_subset(basis, names, invert = FALSE)
```

Arguments

`basis`	input basis.
`names`	basis set elements to keep in the returned object.
`invert`	set to true to return all basis elements except those given in the names argument.

Value

a subset of the input basis.

`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_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_head_dyns` *Return the first scans of a dynamic series.*

Description

Return the first scans of a dynamic series.

Usage

```
get_head_dyns(mrs_data, n = 1)
```

Arguments

`mrs_data`	dynamic MRS data.
`n`	the number of dynamic scans to return.

Value

first scans of a dynamic series.

`get_lcm_cmd` *Print the command to run the LCModel command-line program.*

Description

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

Usage

```
get_lcm_cmd()
```

`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 and individual segmentation maps.

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. Default value is mmand::boxKernel(), use mmand::mnKernel() for a smoothed map.

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_mrs_affine *Generate an affine for nifti generation.*

Description

Generate an affine for nifti generation.

Usage

```
get_mrs_affine(mrs_data, x_pos = 1, y_pos = 1, z_pos = 1)
```

Arguments

`mrs_data`	input data.
`x_pos`	x_position coordinate.
`y_pos`	y_position coordinate.
`z_pos`	z_position coordinate.

Value

affine matrix.

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_spin_num *Return the spin number for a given nucleus.*

Description

Return the spin number for a given nucleus.

Usage

```
get_spin_num(nucleus)
```

Arguments

nucleus	nucleus name, eg "1H".

Value

spin number.

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,
  fd_set = NULL,
  td_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).
`fd_set`	frequency domain data indices to include in the output (default all).
`td_set`	time-domain 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_tail_dyns *Return the last scans of a dynamic series.*

Description

Return the last scans of a dynamic series.

Usage

```
get_tail_dyns(mrs_data, n = 1)
```

Arguments

`mrs_data`	dynamic MRS data.
`n`	the number of dynamic scans to return.

Value

last scans of a dynamic series.

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, method = "poly")
```

Arguments

`mrs_data`	MRS data.
`nstart`	first data point to fit.
`nend`	last data point to fit.
`method`	method for measuring the amplitude, one of "poly", spline" or exp".

Value

array of amplitudes.

`get_tqn_cmd` *Print the command to run the TARQUIN command-line program.*

Description

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

Usage

```
get_tqn_cmd()
```

`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,
  full_name = NULL
)
```

Arguments

`name`	abbreviated name of the molecule.
`chem_shift`	chemical shift of the resonance (PPM).
`nucleus`	nucleus (1H, 31P...).
`scale_factor`	multiplicative scaling factor. Note, this value can be made complex to adjust the phase of the resonance.
`lw`	linewidth in Hz.
`lg`	Lorentz-Gauss lineshape parameter (between 0 and 1).
`full_name`	long name of the molecule (optional).

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.

glm_spec *Perform a GLM analysis of dynamic MRS data in the spectral domain.*

Description

Perform a GLM analysis of dynamic MRS data in the spectral domain.

Usage

```
glm_spec(mrs_data, regressor_df)
```

Arguments

`mrs_data`	single-voxel dynamics MRS data.
`regressor_df`	a data frame containing temporal regressors to be applied to each spectral datapoint.

Value

list of statistical results.

gridplot *Arrange spectral plots in a grid.*

Description

Arrange spectral plots in a grid.

Usage

```
gridplot(x, ...)
```

Arguments

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

gridplot.mrs_data *Arrange spectral plots in a grid.*

Description

Arrange spectral plots in a grid.

Usage

```
## S3 method for class 'mrs_data'
gridplot(
  x,
  rows = NA,
  cols = NA,
  mar = c(0, 0, 0, 0),
  oma = c(3.5, 1, 1, 1),
  bty = "o",
  restore_def_par = TRUE,
  ...
)
```

Arguments

`x`	object of class mrs_data.
`rows`	number of grid rows.
`cols`	number of grid columns.
`mar`	option to adjust the plot margins. See ?par.
`oma`	outer margin area.
`bty`	option to draw a box around the plot. See ?par.
`restore_def_par`	restore default plotting par values after the plot has been made.
`...`	other arguments to pass to the plot method.

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` *HSVD of an mrs_data object.*

Description

HSVD method as 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(mrs_data, comps = 40, irlba = TRUE, max_damp = 10)
```

Arguments

`mrs_data`	mrs_data object to be decomposed.
`comps`	number of Lorentzian components to use for modelling.
`irlba`	option to use irlba SVD (logical).
`max_damp`	maximum allowable damping factor.

Value

basis matrix and signal table.

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,
  max_damp = 10,
  scale = "hz",
  return_model = FALSE
)
```

Arguments

`mrs_data`	MRS data to be filtered.
`xlim`	frequency range to filter, default units are Hz which can be changed to ppm using the "scale" argument.
`comps`	number of Lorentzian components to use for modelling.
`irlba`	option to use irlba SVD (logical).
`max_damp`	maximum allowable damping factor.
`scale`	either "hz" or "ppm" to set the frequency units of xlim.
`return_model`	by default the filtered spectrum is returned. Set return_model to TRUE to return the HSVD model of the data.

Value

filtered data or model depending on the return_model argument.

`hsvd_vec` *HSVD of a complex vector.*

Description

HSVD method as 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_vec(y, fs, comps = 40, irlba = TRUE, max_damp = 0)
```

Arguments

`y`	time domain signal to be filtered as a vector.
`fs`	sampling frequency of y.
`comps`	number of Lorentzian components to use for modelling.
`irlba`	option to use irlba SVD (logical).
`max_damp`	maximum allowable damping factor. Default value of 0 ensures resultant model is damped.

Value

basis matrix and signal table.

`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,
  plot_dim = NULL,
  x_pos = NULL,
  y_pos = NULL,
  z_pos = NULL,
  dyn = 1,
  coil = 1,
  restore_def_par = TRUE,
  y_ticks = NULL,
  vline = NULL,
  hline = NULL,
  legend = FALSE,
  ...
)
```

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.
`plot_dim`	the dimension to display on the y-axis, can be one of: "dyn", "time_sec", x", "y", "z", "coil" or NULL. If NULL (the default) all spectra are collapsed into the dynamic dimension and displayed.
`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 additional red tick marks.
`vline`	draw a vertical line at the value of vline.
`hline`	draw a horizontal line at the value of hline.
`legend`	add a colour bar to the plot using the imagePlot function from the fields package.
`...`	other arguments to pass to the plot method.

`img2kspace_xy` *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

```
img2kspace_xy(mrs_data)
```

Arguments

`mrs_data`	2D MRSI data.

Value

k-space data.

`Imzap` *Complex rounding function taken from complexplus package to reduce the number of spant dependencies.*

Description

Complex rounding function taken from complexplus package to reduce the number of spant dependencies.

Usage

```
Imzap(x, tol = 1e-06)
```

Arguments

`x`	a scalar or vector, real or complex.
`tol`	a tolerance, 10^-6 by default. Prevents possible numerical problems. Can be set to 0 if desired.

`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

See spec_op function for a more complete set of spectral operations.

Usage

```
int_spec(mrs_data, xlim = NULL, freq_scale = "ppm", mode = "re")
```

Arguments

`mrs_data`	MRS data.
`xlim`	spectral range to be integrated (defaults to full range).
`freq_scale`	units of xlim, can be : "ppm", "hz" or "points".
`mode`	spectral mode, can be : "re", "im", "mod" or "cplx".

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.def` *Check if an object is defined, which is the same as being not NULL.*

Description

Check if an object is defined, which is the same as being not NULL.

Usage

```
is.def(x)
```

Arguments

`x`	object to test for being NULL.

Value

logical value.

`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.

`kspace2img_xy` *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

```
kspace2img_xy(mrs_data)
```

Arguments

`mrs_data`	2D MRSI data.

Value

MRSI data in image space.

l2_reg *Perform l2 regularisation artefact suppression.*

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,
  thresh = 0.05,
  b = 1e-11,
  A = NA,
  xlim = NA,
  thresh_xlim = NULL,
  A_append = NULL,
  ret_norms = FALSE
)
```

Arguments

`mrs_data`	input data for artefact suppression.
`thresh`	threshold parameter to extract lipid signals from mrs_data based on the spectral integration of the thresh_xlim region in magnitude mode.
`b`	regularisation parameter.
`A`	set of spectra containing the artefact basis signals. The thresh parameter is ignored when A is specified.
`xlim`	spectral limits in ppm to restrict the reconstruction range. Defaults to the full spectral width.
`thresh_xlim`	spectral limits in ppm to integrate for the threshold map.
`A_append`	additional spectra to append to the A basis.
`ret_norms`	return the residual norm and solution norms.

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 'list'
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.

lofdc — *Correct linear frequency drift.*

Description

Correct linear frequency drift.

Usage

```
lofdc(
  mrs_data,
  max_hz_s = 0.1,
  tr = NULL,
  ret_corr_only = TRUE,
  outlier_thresh = 3,
  xlim = c(4, 0.5),
  order = 1
)
```

Arguments

`mrs_data`	MRS data to be corrected.
`max_hz_s`	the maximum drift rate to search over.
`tr`	mrs_data repetition time.
`ret_corr_only`	return the corrected mrs_data object only.
`outlier_thresh`	threshold to remove outliers.
`xlim`	spectral width (in ppm) to evaluate outliers.
`order`	correction order.

Value

drift corrected mrs_data object.

`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.

make_basis_from_raw — *Make a basis-set object from a directory containing LCModel formatted RAW files.*

Description

Make a basis-set object from a directory containing LCModel formatted RAW files.

Usage

```
make_basis_from_raw(dir_path, ft, fs, ref)
```

Arguments

`dir_path`	path to the directory containing LCModel RAW files. One file per signal.
`ft`	transmitter frequency in Hz.
`fs`	sampling frequency in Hz.
`ref`	reference value for ppm scale.

Value

a basis-set object.

`mask_dyns` *Mask an MRS dataset in the dynamic dimension.*

Description

Mask an MRS dataset in the dynamic dimension.

Usage

```
mask_dyns(mrs_data, mask)
```

Arguments

`mrs_data`	MRS data object.
`mask`	vector of boolean values specifying the dynamics to mask, where a value of TRUE indicates the spectrum should be removed.

Value

masked dataset.

`mask_fit_res` *Mask fit result spectra depending on a vector of bool values.*

Description

Mask fit result spectra depending on a vector of bool values.

Usage

```
mask_fit_res(fit_result, mask_vec, amps_only = FALSE)
```

Arguments

`fit_result`	fit result object to be masked.
`mask_vec`	a Boolean vector with the same number of rows as there are rows in the results table.
`amps_only`	only mask the amplitude and associated error estimate columns.

Value

a masked fit result object.

`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_corners` *Mask the four corners of an MRSI dataset in the x-y plane.*

Description

Mask the four corners of an MRSI dataset in the x-y plane.

Usage

```
mask_xy_corners(mrs_data)
```

Arguments

`mrs_data`	MRS data object.

Value

masked MRS data.

`mask_xy_ellipse` *Mask the voxels outside an elliptical region spanning the MRSI dataset in the x-y plane.*

Description

Mask the voxels outside an elliptical region spanning the MRSI dataset in the x-y plane.

Usage

```
mask_xy_ellipse(mrs_data)
```

Arguments

`mrs_data`	MRS data object.

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, value = NA)
```

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`	the value to set masked data to (usually NA or 0).

Value

masked dataset.

mat2mrs_data

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

Description

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

Usage

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

Arguments

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

Value

mrs_data object.

matexp

Matrix exponential function taken from complexplus package to reduce the number of spant dependencies.

Description

Matrix exponential function taken from complexplus package to reduce the number of spant dependencies.

Usage

```
matexp(x)
```

Arguments

`x`	a square complex matrix.

Value

the matrix exponential of x.

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

Description

Calculate the mean spectrum from an mrs_data object.

Usage

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

Arguments

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

Value

mean mrs_data object.

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, subset = NULL)
```

Arguments

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

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.

`mean_mrs_list` *Return the mean of a list of mrs_data objects.*

Description

Return the mean of a list of mrs_data objects.

Usage

```
mean_mrs_list(mrs_list)
```

Arguments

`mrs_list`	list of mrs_data objects.

Value

mean `mrs_data` object.

`mean_vec_blocks` *Calculate the mean of adjacent blocks in a vector.*

Description

Calculate the mean of adjacent blocks in a vector.

Usage

```
mean_vec_blocks(x, block_size)
```

Arguments

`x`	input vector.
`block_size`	number of adjacent elements to average over.

Value

vector data averaged in blocks.

`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.

`mod_td` *Apply the Modulus operator to the time-domain MRS signal.*

Description

Apply the Modulus operator to the time-domain MRS signal.

Usage

```
mod_td(mrs_data)
```

Arguments

`mrs_data`	MRS data input.

Value

time-domain modulus of input.

`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, collapse = TRUE)
```

Arguments

`mrs_data`	MRS data object or list of MRS data objects.
`collapse`	collapse all other dimensions along the dynamic dimension, eg a 16x16 MRSI grid would be first collapsed across 256 dynamic scans.

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.

`mrs_data_list2bids` *Create a BIDS directory and file structure from a list of mrs_data objects.*

Description

Create a BIDS directory and file structure from a list of mrs_data objects.

Usage

```
mrs_data_list2bids(
  mrs_data_list,
  output_dir,
  runs = 1,
  sessions = 1,
  sub_labels = NULL
)
```

Arguments

`mrs_data_list`	list of mrs_data objects.
`output_dir`	the base directory to create the BIDS structure.
`runs`	number of runs per subject and session.
`sessions`	number of sessions.
`sub_labels`	optional labels for subject level identification.

`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.

`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.

Ntrans — *Return the total number of acquired transients for an MRS dataset.*

Description

Return the total number of acquired transients for an MRS dataset.

Usage

```
Ntrans(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.

`one_page_pdf` *Export a one-page pdf of a single fit result*

Description

Export a one-page pdf of a single fit result

Usage

```
one_page_pdf(fit_res, pdf_out_path, title = NULL)
```

Arguments

`fit_res`	`fit_result` object.
`pdf_out_path`	path to the exported pdf file.
`title`	ouptut title.

`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 = 0.7,
  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(0.63, 0.65, 0.07, 0.42)
)
```

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_inter` *Display an interactive orthographic projection plot of a nifti object.*

Description

Display an interactive orthographic projection plot of a nifti object.

Usage

```
ortho3_inter(
  underlay,
  overlay = NULL,
  xyz = NULL,
  zlim = NULL,
  zlim_ol = NULL,
  alpha = 0.7,
  ...
)
```

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.

`pg_extrap_xy` *Papoulis-Gerchberg (PG) algorithm method for k-space extrapolation.*

Description

PG method as described in: Haupt CI, Schuff N, Weiner MW, Maudsley AA. Removal of lipid artifacts in 1H spectroscopic imaging by data extrapolation. Magn Reson Med. 1996 May;35(5):678-87. Extrapolation is performed to expand k-space coverage by a factor of 2, with the aim to reduce Gibbs ringing.

Usage

```
pg_extrap_xy(
  mrs_data,
  img_mask = NULL,
  kspace_mask = NULL,
  intensity_thresh = 0.15,
  iters = 50
)
```

Arguments

`mrs_data`	MRS data object.
`img_mask`	a boolean matrix of voxels with strong signals to be extrapolated. Must be twice the dimensions of the input data.
`kspace_mask`	a boolean matrix of kspace points that have been sampled. Typically a circle for MRSI, but defaults to the full rectangular area of k-space covered by the input data. Must match the x-y dimensions of the input data.
`intensity_thresh`	used to define img_mask based on the strength of the signal in each voxel. Defaults to intensities greater than 15% of the maximum. Ignored if img_mask is specified as argument.
`iters`	number of iterations to perform.

Value

extrapolated `mrs_data` object.

`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,
  show_grid = TRUE,
  grid_nx = NULL,
  grid_ny = NA,
  invert_fit = 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).
`show_grid`	plot gridlines behind the data (logical). Defaults to TRUE.
`grid_nx`	number of cells of the grid in x and y direction. When NULL the grid aligns with the tick marks on the corresponding default axis (i.e., tickmarks as computed by axTicks). When NA, no grid lines are drawn in the corresponding direction.
`grid_ny`	as above.
`invert_fit`	show the fit result "upside-down"/
`...`	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,
  yaxis_lab = NULL,
  xat = NULL,
  xlabs = TRUE,
  yat = NULL,
  ylabs = TRUE,
  show_grid = TRUE,
  grid_nx = NULL,
  grid_ny = NA,
  col = NULL,
  alpha = NULL,
  bl_lty = NULL,
  hline = NULL,
  hline_lty = 2,
  hline_col = "red",
  vline = NULL,
  vline_lty = 2,
  vline_col = "red",
  ...
)
```

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.
`yaxis_lab`	y-axis label.
`xat`	x-axis tick label values.
`xlabs`	x-axis tick labels.
`yat`	y-axis tick label values.
`ylabs`	y-axis tick labels.
`show_grid`	plot gridlines behind the data (logical). Defaults to TRUE.
`grid_nx`	number of cells of the grid in x and y direction. When NULL the grid aligns with the tick marks on the corresponding default axis (i.e., tickmarks as computed by axTicks). When NA, no grid lines are drawn in the corresponding direction.
`grid_ny`	as above.
`col`	set the line colour, eg col = rgb(0.5, 0.5, 0.5).
`alpha`	set the line transparency, eg alpha = 0.5 is 50% transparency. Overrides any transparency levels set by col.
`bl_lty`	linetype for the y = 0 baseline trace. A default value NULL results in no baseline being plotted.
`hline`	add a horizontal line at the specified value.

`hline_lty`	linetype for the horizontal line.
`hline_col`	colour for the horizontal line.
`vline`	add a vertical line at the specified value.
`vline_lty`	linetype for the vertical line.
`vline_col`	colour for the vertical line.
`...`	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_reg — *Plot regressors as an image.*

Description

Plot regressors as an image.

Usage

```
plot_reg(regressor_df)
```

Arguments

`regressor_df`	input regressor data frame.

`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,
  fd = TRUE
)
```

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.
`fd`	display data in the frequency-domain (default), or time-domain (logical).

`plot_spec_sd` *Plot the spectral standard deviation.*

Description

Plot the spectral standard deviation.

Usage

```
plot_spec_sd(mrs_data, xlim = NULL, scale_sd = 1.96, ...)
```

Arguments

`mrs_data`	MRS data to be plotted.
`xlim`	plotting limits in ppm.
`scale_sd`	scaling factor for the standard deviation trace.
`...`	other arguments passed to the stackplot function.

`plot_voi_overlay` *Plot a volume as an image overlay.*

Description

Plot a volume as an image overlay.

Usage

```
plot_voi_overlay(mri, voi, export_path = NULL, zlim = NULL, ...)
```

Arguments

`mri`	image data as a nifti object or path to data file.
`voi`	volume data as a nifti object or path to data file.
`export_path`	optional path to save the image in png format.
`zlim`	underlay intensity limits.
`...`	additional arguments to the ortho3 function.

`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(mri_seg, voi, export_path = NULL, ...)
```

Arguments

`mri_seg`	segmented brain volume as a nifti object.
`voi`	volume data as a nifti object.
`export_path`	optional path to save the image in png format.
`...`	additional arguments to the ortho3 function.

`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.

`precomp` *Save function results to file and load on subsequent calls to avoid repeat computation.*

Description

Save function results to file and load on subsequent calls to avoid repeat computation.

Usage

```
precomp(file, fun, ...)
```

Arguments

`file`	file name to write the results.
`fun`	function to run.
`...`	arguments to be passed to fun.

`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,
  sp_N = 2,
  sp_deg = 3,
  max_t = 0.2,
  basis_type = "poly",
  rescale_output = TRUE,
  phase_corr = TRUE,
  ret_corr_only = TRUE,
  zero_freq_shift_t0 = FALSE,
  remove_freq_outliers = FALSE,
  freq_outlier_thresh = 3,
  remove_phase_outliers = FALSE,
  phase_outlier_thresh = 3,
  remove_amp_outliers = FALSE,
  amp_outlier_thresh = 3
)
```

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.
`sp_N`	number of spline functions, note the true number will be sp_N + sp_deg.
`sp_deg`	degree of spline functions.
`max_t`	truncate the FID when longer than max_t to reduce time taken, set to NULL to use the entire FID.

`basis_type`	may be one of "poly" or "spline".
`rescale_output`	rescale the bl_matched_spec and bl output to improve consistency between dynamic scans.
`phase_corr`	apply phase correction (in addition to frequency). TRUE by default.
`ret_corr_only`	return the corrected mrs_data object only.
`zero_freq_shift_t0`	perform a linear fit to the frequency shifts and set the (linearly modeled) shift to be 0 Hz for the first dynamic scan.
`remove_freq_outliers`	remove dynamics based on their frequency shift.
`freq_outlier_thresh`	threshold to remove frequency outliers.
`remove_phase_outliers`	remove dynamics based on their phase shift.
`phase_outlier_thresh`	threshold to remove phase outliers.
`remove_amp_outliers`	remove dynamics based on their amplitude change.
`amp_outlier_thresh`	threshold to remove amplitude outliers.

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(), sort_basis = TRUE)
```

Arguments

`basis_file`	path to basis file.
`ref`	assumed ppm reference value.
`sort_basis`	sort the basis set based on signal names.

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, extra = NULL, verbose = FALSE)
```

Arguments

`dir`	data directory path.
`extra`	an optional data frame to provide additional variables for use in subsequent analysis steps, eg id or grouping variables.
`verbose`	output extra information to the console.

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, extra = NULL, verbose = FALSE)
```

Arguments

`dir`	data directory path.
`extra`	an optional data frame to provide additional variables for use in subsequent analysis steps, eg id or grouping variables.
`verbose`	output extra information to the console.

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_fid = FALSE,
  omit_svs_ref_scans = TRUE,
  verbose = FALSE,
  extra = NULL
)
```

Arguments

`fname`	filename of the dpt format MRS data.
`format`	string describing the data format. Must be one of the following : "spar_sdat", "rda", "dicom", "twix", "pfile", "list_data", "paravis", "dpt", "lcm_raw", "rds", "nifti", "varian", "jmrui_txt". 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_fid`	export all data points, including those before the start of the FID (default = FALSE), TWIX format only.
`omit_svs_ref_scans`	remove any reference scans sometimes saved in SVS twix data (default = TRUE).
`verbose`	print data file information (default = FALSE).
`extra`	an optional data frame to provide additional variables for use in subsequent analysis steps, eg id or grouping variables.

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_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_pulse_ascii` *Read an ASCII formatted pulse file.*

Description

Read an ASCII formatted pulse file.

Usage

```
read_pulse_ascii(fname, deg2rad = TRUE)
```

Arguments

`fname`	ASCII formatted pulse file path.
`deg2rad`	convert phase values stored in degrees to radians.

Value

pulse waveform and header.

`read_pulse_bruker` *Read a Bruker formatted pulse file*

Description

Read a Bruker formatted pulse file

Usage

```
read_pulse_bruker(fname)
```

Arguments

`fname`	Bruker formatted pulse file path.

Value

pulse waveform and header.

`read_pulse_pta` *Read a .pta formatted pulse file compatible with Siemens PulseTool.*

Description

Read a .pta formatted pulse file compatible with Siemens PulseTool.

Usage

```
read_pulse_pta(fname)
```

Arguments

`fname`	pta formatted pulse file path.

Value

pulse waveform and header.

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

Description

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

Usage

```
read_siemens_txt_hdr(input, version = "vd", verbose, offset = 0)
```

Arguments

`input`	file name to read or raw data.
`version`	software version, can be "vb" or "vd".
`verbose`	print information to the console.
`offset`	offset to begin searching for the text header.

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)
```

`recon_imag` *Reconstruct complex time-domain data from the real part of frequency-domain data.*

Description

Reconstruct complex time-domain data from the real part of frequency-domain data.

Usage

```
recon_imag(mrs_data)
```

Arguments

`mrs_data`	MRS data.

Value

reconstructed MRS data.

`recon_imag_vec` *Reconstruct complex time-domain data from the real part of frequency-domain data.*

Description

Reconstruct complex time-domain data from the real part of frequency-domain data.

Usage

```
recon_imag_vec(data)
```

Arguments

`data`	data points in the frequency domain.

Value

reconstructed signal.

`recon_twix_2d_mrsi` *Reconstruct 2D MRSI data from a twix file loaded with read_mrs.*

Description

Reconstruct 2D MRSI data from a twix file loaded with read_mrs.

Usage

```
recon_twix_2d_mrsi(twix_mrs)
```

Arguments

`twix_mrs`	raw dynamic data.

Value

reconstructed data.

`rectangular_mask` *Create a rectangular mask stored as a matrix of logical values.*

Description

Create a rectangular mask stored as a matrix of logical values.

Usage

```
rectangular_mask(xN, yN, x0, y0, xw, yw, angle)
```

Arguments

`xN`	number of pixels in the x dimension.
`yN`	number of pixels in the y dimension.
`x0`	centre of rectangle in the x direction in units of pixels.
`y0`	centre of rectangle in the y direction in units of pixels.
`xw`	width in the x direction in units of pixels.
`yw`	width in the y direction in units of pixels.
`angle`	angle of rotation in degrees.

Value

logical mask matrix with dimensions fov_yN x fov_xN.

`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,
  warn = TRUE
)
```

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.
`warn`	print a warning when the data dimensions do not change.

Value

replicated data object.

resample_basis — *Resample a basis-set to match a mrs_data acquisition.*

Description

Resample a basis-set to match a mrs_data acquisition.

Usage

```
resample_basis(basis, mrs_data, ref_freq_match = TRUE)
```

Arguments

`basis`	the basis to be resampled.
`mrs_data`	the mrs_data to match the number of data points and sampling frequency.
`ref_freq_match`	apply a frequency shift to the basis to match the reference frequency (usually 4.65 or 4.68) of the mrs_data.

Value

resampled basis set 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, interp = 3L)
```

Arguments

`mri`	nifti object to be resliced.
`mrs`	mrs_data object for the target orientation.
`interp`	interpolation parameter, see nifyreg.linear definition.

Value

resliced imaging data.

`reson_table2mrs_data` *Generate mrs_data from a table of single Lorentzian resonances.*

Description

Generate mrs_data from a table of single Lorentzian resonances.

Usage

```
reson_table2mrs_data(
  reson_table,
  acq_paras = def_acq_paras(),
  back_extrap_pts = 0
)
```

Arguments

`reson_table`	as produced by the hsvd function.
`acq_paras`	list of acquisition parameters. See
`back_extrap_pts`	number of data points to back extrapolate `def_acq_paras`

Value

mrs_data object.

`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 (Gaussian 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` *Apply water reference scaling to a fitting results object to yield metabolite quantities in millimolar (mM) units (mol / kg of tissue water).*

Description

Note, this function assumes the volume contains a homogeneous voxel, eg pure WM, GM or CSF. Also note that in the case of a homogeneous voxel the relative densities of MR-visible water (eg GM=0.78, WM=0.65, and CSF=0.97) cancel out and don't need to be considered. Use scale_amp_molal_pvc for volumes containing multiple compartments. Details of this method can be found in "Use of tissue water as a concentration reference for proton spectroscopic imaging" by Gasparovic et al MRM 2006 55(6):1219-26.

Usage

```
scale_amp_molal(
  fit_result,
  ref_data,
  te,
  tr,
  water_t1,
  water_t2,
  metab_t1,
  metab_t2,
  ...
)
```

Arguments

`fit_result`	result object generated from fitting.
`ref_data`	water reference MRS data object.
`te`	the MRS TE in seconds.
`tr`	the MRS TR in seconds.
`water_t1`	assumed water T1 value.
`water_t2`	assumed water T2 value.
`metab_t1`	assumed metabolite T1 value.
`metab_t2`	assumed metabolite T2 value.
`...`	additional arguments to get_td_amp function.

Value

A `fit_result` object with a rescaled results table.

`scale_amp_molal_pvc` *Apply water reference scaling to a fitting results object to yield metabolite quantities in millimolar (mM) units (mol / kg of tissue water).*

Description

Details of this method can be found in "Use of tissue water as a concentration reference for proton spectroscopic imaging" by Gasparovic et al MRM 2006 55(6):1219-26. 1.5 Tesla relaxation assumptions are taken from this paper. For 3 Tesla data, relaxation assumptions are taken from "NMR relaxation times in the human brain at 3.0 Tesla" by Wansapura et al J Magn Reson Imaging 1999 9(4):531-8.

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 expressed as percentages. For example, a voxel containing 100% white matter tissue would use : p_vols = c(WM = 100, GM = 0, CSF = 0).
`te`	the MRS TE in seconds.
`tr`	the MRS TR in seconds.
`...`	additional arguments to get_td_amp function.

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 of tissue).*

Description

See the LCModel manual (section 10.2) on water-scaling for details on the assumptions and relevant references. Use this type of concentration scaling to compare fit results with LCModel and TARQUIN defaults. Otherwise scale_amp_molal_pvc is generally the preferred method.

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). Assumes water T2 of 80ms and a TE = 30 ms. exp(-30ms / 80ms) ~ 0.7.
`w_conc`	assumed water concentration (default = 35880). Default value corresponds to typical white matter. Set to 43300 for gray matter, and 55556 for phantom measurements.
`...`	additional arguments to get_td_amp function.

Value

a `fit_result` object with a rescaled results table.

scale_amp_molar2molal_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

```
scale_amp_molar2molal_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 expressed as percentages. For example, a voxel containing 100% white matter tissue would use : p_vols = c(WM = 100, GM = 0, CSF = 0).
`te`	the MRS TE.
`tr`	the MRS TR.

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, use_mean_value = FALSE)
```

Arguments

`fit_result`	a result object generated from fitting.
`name`	the signal name to use as a denominator (usually, "tCr" or "tNAA").
`use_mean_value`	scales the result by the mean of the signal when set to TRUE.

Value

a `fit_result` object with a rescaled results table.

`scale_amp_ratio_value` *Scale fitted amplitudes to a ratio of signal amplitude.*

Description

Scale fitted amplitudes to a ratio of signal amplitude.

Usage

```
scale_amp_ratio_value(fit_result, value)
```

Arguments

`fit_result`	a result object generated from fitting.
`value`	the number use as a denominator.

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.
`...`	additional arguments to get_td_amp function.

Value

a `fit_result` object with a rescaled results table.

`scale_basis_amp` *Scale a basis object by a scalar.*

Description

Scale a basis object by a scalar.

Usage

```
scale_basis_amp(basis, amp)
```

Arguments

`basis`	basis_set object to be scaled.
`amp`	multiplicative factor with length 1.

Value

basis_set object multiplied by the amplitude scale factor.

`scale_basis_from_singlet` *Scale a basis-set to be consistent with spant assumptions for water scaling.*

Description

For correct water scaling, spant assumes the time-domain amplitude (t = 0) for a single proton is 0.5. Internally simulated basis-sets will be correctly scaled, however imported basis-sets should be assumed to be un-scaled and this function should be used. Note that the singlet specified should only contain one resonance, and that any additional signals (eg TSP or residual water) will result in incorrect scaling. Therefore, only simulated basis sets are appropriate for use with this function.

Usage

```
scale_basis_from_singlet(basis, name, protons)
```

Arguments

`basis`	basis set to be scaled.
`name`	the name of the singlet to be used as a scaling reference.
`protons`	the number of MRS visible protons contributing to the singlet resonance.

Value

a scaled basis.

scale_mrs_amp *Scale an mrs_data object by a scalar or vector or amplitudes.*

Description

Scale an mrs_data object by a scalar or vector or amplitudes.

Usage

```
scale_mrs_amp(mrs_data, amp)
```

Arguments

mrs_data	data to be scaled.
amp	multiplicative factor, must have length equal to 1 or Nspec(mrs_data).

Value

mrs_data object multiplied by the amplitude scale factor.

scale_spec *Scale mrs_data to a spectral region.*

Description

Scale mrs_data to a spectral region.

Usage

```
scale_spec(
  mrs_data,
  xlim = NULL,
  operator = "sum",
  freq_scale = "ppm",
  mode = "re",
  mean_dyns = NULL,
  ret_scale_factor = FALSE
)
```

Arguments

`mrs_data`	MRS data.
`xlim`	spectral range to be integrated (defaults to full range).
`operator`	can be "sum" (default), "mean", "l2", "max", "min" or "max-min".
`freq_scale`	units of xlim, can be : "ppm", "Hz" or "points".
`mode`	spectral mode, can be : "re", "im", "mod" or "cplx".
`mean_dyns`	mean the dynamic scans before applying the operator. The same scaling value will be applied to each individual dynamic.
`ret_scale_factor`	option to return the scaling factor in addition to the scaled data.

Value

normalised data.

`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_2d_shaped` *PRESS sequence with shaped refocusing pulses.*

Description

PRESS sequence with shaped refocusing pulses.

Usage

```
seq_press_2d_shaped(
  spin_params,
  ft,
  ref,
  TE1 = 0.01,
  TE2 = 0.02,
  pulse_file,
  pulse_dur,
  pulse_file_format,
  refoc_flip_angle = 180,
  xy_pulse_ppm = NULL,
  resamp = TRUE,
  fs_resamp = 1e-04
)
```

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.
`pulse_file`	path to refocusing pulse file.
`pulse_dur`	refocusing pulse duration.
`pulse_file_format`	file format for the refocusing pulse.
`refoc_flip_angle`	refocusing pulse flip angle in degrees (defaults to 180).
`xy_pulse_ppm`	a vector of ppm values for the offset of each sub-simulation.
`resamp`	option to resample the pulse shape.
`fs_resamp`	sampling frequency (Hz) to resample.

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, nuc = "1H", acq_delay = 0)
```

Arguments

spin_params	spin system definition.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.
nuc	acquisition nucleus.
acq_delay	delay between excitation and acquisition.

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, nuc = "1H", TE = 0.03)
```

Arguments

`spin_params`	spin system definition.
`ft`	transmitter frequency in Hz.
`ref`	reference value for ppm scale.
`nuc`	acquisition nucleus.
`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, amp_scale = 2)
```

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.
`amp_scale`	amplitude scaling factor. Set to 2 (default) to ensure correct scaling for water reference scaling. Set to 1 to maintain the inherent loss of signal associated with STEAM.

Value

list of resonance amplitudes and frequencies.

---

seq_steam_ideal_cof   *STEAM sequence with ideal pulses and coherence order filtering to simulate gradient crushers.*

Description

See Landheer et al NMR Biomed 2021 34(5):e4129 and Landheer et al MRM 2019 Apr;81(4):2209-2222 for more details on the coherence order filtering method.

Usage

```
seq_steam_ideal_cof(spin_params, ft, ref, TE = 0.03, TM = 0.02, amp_scale = 2)
```

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.
`amp_scale`	amplitude scaling factor. Set to 2 (default) to ensure correct scaling for water reference scaling. Set to 1 to maintain the inherent loss of signal associated with STEAM.

Value

list of resonance amplitudes and frequencies.

---

seq_steam_ideal_young   *STEAM sequence with ideal pulses using the z-rotation gradient simulation method described by Young et al JMR 140, 146-152 (1999).*

Description

STEAM sequence with ideal pulses using the z-rotation gradient simulation method described by Young et al JMR 140, 146-152 (1999).

Usage

```
seq_steam_ideal_young(
  spin_params,
  ft,
  ref,
  TE = 0.03,
  TM = 0.02,
  amp_scale = 2
)
```

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.
`amp_scale`	amplitude scaling factor. Set to 2 (default) to ensure correct scaling for water reference scaling. Set to 1 to maintain the inherent loss of signal associated with STEAM.

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"),
  nuc = getOption("spant.nuc")
)
```

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.
`nuc`	resonant nucleus.

`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), lg = 1, mask_narrow = TRUE)
```

Arguments

`mrs_data`	data in.
`lw`	target linewidth in units of ppm.
`xlim`	region to search for peaks to obtain a linewidth estimate.
`lg`	Lorentz-Gauss lineshape parameter.
`mask_narrow`	masks spectra where the requested linewidth is too narrow, if set FALSE the spectra are not changed.

Value

line-broadened data.

`set_mask_xy_mat` *Set the masked voxels in a 2D MRSI dataset to given spectrum.*

Description

Set the masked voxels in a 2D MRSI dataset to given spectrum.

Usage

```
set_mask_xy_mat(mrs_data, mask, mask_mrs_data)
```

Arguments

`mrs_data`	MRSI data object.
`mask`	matrix of boolean values specifying the voxels to set, where a value of TRUE indicates the voxel should be set to mask_mrs_data.
`mask_mrs_data`	the spectral data to be assigned to the masked voxels.

Value

updated dataset.

`set_Ntrans` *Set the number of transients for an mrs_data object.*

Description

Set the number of transients for an mrs_data object.

Usage

```
set_Ntrans(mrs_data, n_trans)
```

Arguments

`mrs_data`	MRS data.
`n_trans`	number of acquired transients.

`set_precomp_mode` *Set the precompute mode.*

Description

Set the precompute mode.

Usage

```
set_precomp_mode(mode = NA)
```

Arguments

`mode`	can be one of: "default", "overwrite", "clean" or "disabled".

`set_precomp_verbose` *Set the verbosity of the precompute function.*

Description

Set the verbosity of the precompute function.

Usage

```
set_precomp_verbose(verbose = NA)
```

Arguments

`verbose`	can be TRUE or FALSE.

`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.

`set_tr` *Set the repetition time of an MRS dataset.*

Description

Set the repetition time of an MRS dataset.

Usage

```
set_tr(mrs_data, tr)
```

Arguments

`mrs_data`	MRS data.
`tr`	repetition time in seconds.

Value

updated mrs_data set.

`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.

`shift_basis` *Apply frequency shifts to basis set signals.*

Description

Apply frequency shifts to basis set signals.

Usage

```
shift_basis(basis, shifts)
```

Arguments

`basis`	the basis to apply the shift to.
`shifts`	a vector of frequency shifts to apply in ppm units. Must be the same length as there are basis elements, or one value to be applied to all elements.

Value

modified basis set object.

`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,
  verbose = FALSE,
  ...
)
```

Arguments

`mol_list`	list of `mol_parameter` objects. Alternatively, a character vector matching molecules may also be provided. Use the get_mol_names function for a full list of molecules.
`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.
`verbose`	output simulation progress and timings.
`...`	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_mm_lip_lcm

Simulate a macromolecular and lipid basis-set suitable for 1H brain MRS analysis.

Description

Simulate a macromolecular and lipid basis-set suitable for 1H brain MRS analysis.

Usage

```
sim_basis_mm_lip_lcm(acq_paras = def_acq_paras())
```

Arguments

`acq_paras`	list of acquisition parameters or an mrs_data object. See `def_acq_paras`

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_v2",
  pul_seq = seq_slaser_ideal,
  xlim = c(0.5, 4.2),
  full_output = FALSE,
  amps = NULL,
  basis_lb = NULL,
  zero_lip_mm = FALSE,
  remove_lip_mm = FALSE,
  ...
)
```

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.
`basis_lb`	apply additional Gaussian line-broadening to the basis (Hz).
`zero_lip_mm`	zero the amplitudes of any lipid or macromolecular components based on their name starting with "MM" or "Lip".
`remove_lip_mm`	remove any lipid or macromolecular basis components based on their name starting with "MM" or "Lip".
`...`	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 an mrs_data object containing simulated Gaussian noise.*

Description

Simulate an 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(),
  fp_scale = TRUE,
  back_extrap_pts = 0,
  sum_resonances = TRUE
)
```

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`
`fp_scale`	multiply the first data point by 0.5.
`back_extrap_pts`	number of data points to back extrapolate.
`sum_resonances`	sum all resonances (default is TRUE), otherwise return a dynamic mrs_data object.

Value

MRS data object.

Examples

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

`sim_th_excit_profile` *Simulate an ideal pulse excitation profile by smoothing a top-hat function with a Gaussian.*

Description

Simulate an ideal pulse excitation profile by smoothing a top-hat function with a Gaussian.

Usage

```
sim_th_excit_profile(bw = 1500, sigma = 50, fa = 180)
```

Arguments

`bw`	top-hat bandwidth (Hz).
`sigma`	Gaussian width smoothing parameter (Hz).
`fa`	intended flip angle of the pulse.

Value

data frame containing the frequency scale, excitation profile and corresponding flip-angles.

`sim_zero` *Simulate an mrs_data object containing complex zero valued samples.*

Description

Simulate an mrs_data object containing complex zero valued samples.

Usage

```
sim_zero(fs = def_fs(), ft = def_ft(), N = def_N(), ref = def_ref(), dyns = 1)
```

Arguments

`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.

Value

mrs_data object.

smooth_dyns	*Smooth data across the dynamic dimension with a Gaussian kernel.*

Description

Smooth data across the dynamic dimension with a Gaussian kernel.

Usage

```
smooth_dyns(mrs_data, sigma)
```

Arguments

`mrs_data`	data to be smoothed.
`sigma`	standard deviation of the underlying Gaussian kernel in seconds.

Value

smoothed mrs_data object.

sort_basis	*Sort the basis-set elements alphabetically.*

Description

Sort the basis-set elements alphabetically.

Usage

```
sort_basis(basis)
```

Arguments

`basis`	input basis.

Value

sorted basis.

`spant_abfit_benchmark` *Simulate and fit some spectra with ABfit for benchmarking purposes. Basic timing and performance metrics will be printed.*

Description

Simulate and fit some spectra with ABfit for benchmarking purposes. Basic timing and performance metrics will be printed.

Usage

```
spant_abfit_benchmark(noise_reps = 10, return_res = FALSE, opts = abfit_opts())
```

Arguments

`noise_reps`	number of spectra to fit with differing noise samples.
`return_res`	return a list of fit_result objects.
`opts`	ABfit options structure.

`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.

spant_simulation_benchmark

Simulate a typical metabolite basis set for benchmarking. Timing metrics will be printed on completion.

Description

Simulate a typical metabolite basis set for benchmarking. Timing metrics will be printed on completion.

Usage

```
spant_simulation_benchmark(sim_reps = 10, N = 1024)
```

Arguments

sim_reps	number of times to simulate the basis set.
N	number of FID data points to simulate.

spec_decomp

Decompose an mrs_data object into white and gray matter spectra.

Description

An implementation of the method published by Goryawala et al MRM 79(6) 2886-2895 (2018). "Spectral decomposition for resolving partial volume effects in MRSI".

Usage

```
spec_decomp(mrs_data, wm, gm, norm_fractions = TRUE)
```

Arguments

mrs_data	data to be decomposed into white and gray matter spectra.
wm	vector of white matter contributions to each voxel.
gm	vector of gray matter contributions to each voxel.
norm_fractions	option to normalise the wm, gm vectors for each voxel.

Value

a list of two mrs_data objects corresponding to the two tissue types.

spec_op — *Perform a mathematical operation on a spectral region.*

Description

Perform a mathematical operation on a spectral region.

Usage

```
spec_op(
  mrs_data,
  xlim = NULL,
  operator = "sum",
  freq_scale = "ppm",
  mode = "re"
)
```

Arguments

`mrs_data`	MRS data.
`xlim`	spectral range to be integrated (defaults to full range).
`operator`	can be "sum" (default), "mean", "l2", "max", "min" or "max-min".
`freq_scale`	units of xlim, can be : "ppm", "hz" or "points".
`mode`	spectral mode, can be : "re", "im", "mod" or "cplx".

Value

an array of integral values.

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, precomp_jc_H = NULL, precomp_Iz = NULL)
```

Arguments

`spin_params`	an object describing the spin system properties.
`ft`	transmitter frequency in Hz.
`ref`	reference value for ppm scale.
`precomp_jc_H`	use a precomputed J-coupling H matrix to save time.
`precomp_Iz`	use precomputed Iz matrices to save time.

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.

`ssp` *Signal space projection method for lipid suppression.*

Description

Signal space projection method as described in: Tsai SY, Lin YR, Lin HY, Lin FH. Reduction of lipid contamination in MR spectroscopy imaging using signal space projection. Magn Reson Med 2019 Mar;81(3):1486-1498.

Usage

```
ssp(mrs_data, comps = 5, xlim = c(1.5, 0.8))
```

Arguments

`mrs_data`	MRS data object.
`comps`	the number of spatial components to use.
`xlim`	spectral range (in ppm) covering the lipid signals.

Value

lipid suppressed `mrs_data` 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,
  show_grid = TRUE,
  grid_nx = NULL,
  grid_ny = NA,
  invert_fit = FALSE,
  ...
)
```

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.
`show_grid`	plot gridlines behind the data (logical). Defaults to TRUE.
`grid_nx`	number of cells of the grid in x and y direction. When NULL the grid aligns with the tick marks on the corresponding default axis (i.e., tickmarks as computed by axTicks). When NA, no grid lines are drawn in the corresponding direction.
`grid_ny`	as above.
`invert_fit`	show the fit result "upside-down"/
`...`	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,
  alpha = NULL,
  x_offset = 0,
  y_offset = 0,
  plot_dim = NULL,
  x_pos = NULL,
  y_pos = NULL,
  z_pos = NULL,
  dyn = 1,
  coil = 1,
  bty = NULL,
  labels = NULL,
  lab_cex = 1,
  bl_lty = NULL,
  restore_def_par = TRUE,
  show_grid = NULL,
  grid_nx = NULL,
  grid_ny = NA,
  lwd = NULL,
  vline = NULL,
  vline_lty = 2,
  vline_col = "red",
  mar = 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".
`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).
`alpha`	set the line transparency, eg alpha = 0.5 is 50% transparency. Overrides any transparency levels set by col.
`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.
`plot_dim`	the dimension to display on the y-axis, can be one of: "dyn", "x", "y", "z", "coil" or NULL. If NULL (the default) all spectra will be collapsed into the dynamic dimension and displayed.
`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.
`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.
`show_grid`	plot gridlines behind the data (logical). Defaults to TRUE.
`grid_nx`	number of cells of the grid in x and y direction. When NULL the grid aligns with the tick marks on the corresponding default axis (i.e., tickmarks as computed by axTicks). When NA, no grid lines are drawn in the corresponding direction.
`grid_ny`	as above.
`lwd`	plot linewidth.
`vline`	x-value to draw a vertical line.
`vline_lty`	linetype for the vertical line.
`vline_col`	colour for the vertical line.
`mar`	option to adjust the plot margins. See ?par.
`...`	other arguments to pass to the matplot method.

`sub_first_dyn` *Subtract the first dynamic spectrum from a dynamic series.*

Description

Subtract the first dynamic spectrum from a dynamic series.

Usage

```
sub_first_dyn(mrs_data, scale = 1)
```

Arguments

`mrs_data`	dynamic MRS data.
`scale`	scale factor for the first spectrum.

Value

subtracted data.

`sub_mean_dyns` *Subtract the mean dynamic spectrum from a dynamic series.*

Description

Subtract the mean dynamic spectrum from a dynamic series.

Usage

```
sub_mean_dyns(mrs_data, scale = 1)
```

Arguments

`mrs_data`	dynamic MRS data.
`scale`	scale factor for the mean spectrum.

Value

subtracted data.

`sub_median_dyns` *Subtract the median dynamic spectrum from a dynamic series.*

Description

Subtract the median dynamic spectrum from a dynamic series.

Usage

```
sub_median_dyns(mrs_data, scale = 1)
```

Arguments

`mrs_data`	dynamic MRS data.
`scale`	scale factor for the medium spectrum.

Value

subtracted data.

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.

`sum_mrs` *Sum two mrs_data objects.*

Description

Sum two mrs_data objects.

Usage

```
sum_mrs(a, b, force = FALSE)
```

Arguments

`a`	first mrs_data object to be summed.
`b`	second mrs_data object to be summed.
`force`	set to TRUE to force mrs_data objects to be summed, even if they are in different time/frequency domains.

Value

a + b

`sum_mrs_list` *Return the sum of a list of mrs_data objects.*

Description

Return the sum of a list of mrs_data objects.

Usage

```
sum_mrs_list(mrs_list)
```

Arguments

`mrs_list`	list of mrs_data objects.

Value

sum `mrs_data` object.

svs_1h_brain_analysis *Standard SVS 1H brain analysis pipeline.*

Description

Standard SVS 1H brain analysis pipeline.

Usage

```
svs_1h_brain_analysis(
  metab,
  basis = NULL,
  w_ref = NULL,
  mri_seg = NULL,
  mri = NULL,
  output_dir = NULL,
  extra = NULL,
  decimate = NULL,
  rats_corr = TRUE,
  ecc = FALSE,
  comb_dyns = TRUE,
  hsvd_filt = FALSE,
  scale_amps = TRUE,
  te = NULL,
  tr = NULL,
  preproc_only = FALSE,
  method = "ABFIT",
  opts = NULL
)
```

Arguments

`metab`	filepath or mrs_data object containing MRS metabolite data.
`basis`	basis set object to use for analysis.
`w_ref`	filepath or mrs_data object containing MRS water reference data.
`mri_seg`	filepath or nifti object containing segmented MRI data.
`mri`	filepath or nifti object containing anatomical MRI data.
`output_dir`	directory path to output fitting results.
`extra`	data.frame with one row containing additional information to be attached to the fit results table.
`decimate`	option to decimate the input data by a factor of two. The default value of NULL does not perform decimation unless the spectral width is greater than 20 PPM.
`rats_corr`	option to perform rats correction, defaults to TRUE.
`ecc`	option to perform water reference based eddy current correction, defaults to FALSE.

`comb_dyns`	option to combine dynamic scans, defaults to TRUE.
`hsvd_filt`	option to apply hsvd water removal, defaults to FALSE.
`scale_amps`	option to scale metabolite amplitude estimates, defaults to TRUE.
`te`	metabolite mrs data echo time in seconds.
`tr`	metabolite mrs data repetition time in seconds.
`preproc_only`	only perform the preprocessing steps and omit fitting. The preprocessed metabolite data will be returned in this case.
`method`	analysis method to use, see fit_mrs help.
`opts`	options to pass to the analysis method.

Value

a fit_result or mrs_data object depending on the preproc_only option.

`svs_1h_brain_analysis_dev`

Standard SVS 1H brain analysis pipeline.

Description

Note this function is still under development and liable to changes.

Usage

```
svs_1h_brain_analysis_dev(
  metab,
  w_ref = NULL,
  output_dir = NULL,
  basis = NULL,
  p_vols = NULL,
  append_basis = NULL,
  remove_basis = NULL,
  dfp_corr = FALSE,
  omit_bad_dynamics = FALSE,
  te = NULL,
  tr = NULL,
  output_ratio = "tCr",
  ecc = FALSE,
  abfit_opts = NULL,
  verbose = FALSE
)
```

Arguments

`metab`	filepath or mrs_data object containing MRS metabolite data.
`w_ref`	filepath or mrs_data object containing MRS water reference data.
`output_dir`	directory path to output fitting results.
`basis`	precompiled basis set object to use for analysis.
`p_vols`	a numeric vector of partial volumes expressed as percentages. Defaults to 100% white matter. A voxel containing 100% gray matter tissue would use : p_vols = c(WM = 0, GM = 100, CSF = 0).
`append_basis`	names of extra signals to add to the default basis. Eg append_basis = c("peth", "cit"). Cannot be used with precompiled basis sets.
`remove_basis`	names of signals to remove from the basis. Cannot be used with precompiled basis sets.
`dfp_corr`	perform dynamic frequency and phase correction using the RATS method.
`omit_bad_dynamics`	detect and remove bad dynamics.
`te`	metabolite mrs data echo time in seconds. If not supplied this will be guessed from the metab data file.
`tr`	metabolite mrs data repetition time in seconds. If not supplied this will be guessed from the metab data file.
`output_ratio`	optional string to specify a metabolite ratio to output. Defaults to "tCr" and multiple metabolites may be specified for multiple outputs. Set as NULL to omit.
`ecc`	option to perform water reference based eddy current correction, defaults to FALSE.
`abfit_opts`	options to pass to ABfit.
`verbose`	output potentially useful information.

Examples

```
metab <- system.file("extdata", "philips_spar_sdat_WS.SDAT",
                     package = "spant")
w_ref <- system.file("extdata", "philips_spar_sdat_W.SDAT",
                     package = "spant")
## Not run:
fit_result <- svs_1h_brain_analysis(metab, w_ref, "fit_res_dir")

## End(Not run)
```

svs_1h_brain_batch_analysis

Batch interface to the standard SVS 1H brain analysis pipeline.

Description

Batch interface to the standard SVS 1H brain analysis pipeline.

Usage

```
svs_1h_brain_batch_analysis(
  metab_list,
  w_ref_list = NULL,
  mri_seg_list = NULL,
  mri_list = NULL,
  output_dir_list = NULL,
  extra = NULL,
  ...
)
```

Arguments

`metab_list`	list of file paths or mrs_data objects containing MRS metabolite data.
`w_ref_list`	list of file paths or mrs_data objects containing MRS water reference data.
`mri_seg_list`	list of file paths or nifti objects containing segmented MRI data.
`mri_list`	list of file paths or nifti objects containing anatomical MRI data.
`output_dir_list`	list of directory paths to output fitting results.
`extra`	a data frame with the same number of rows as metab_list, containing additional information to be attached to the fit results table.
`...`	additional options to be passed to the svs_1h_brain_analysis function.

Value

a list of fit_result objects.

`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.

te *Return the echo time of an MRS dataset.*

Description

Return the echo time of an MRS dataset.

Usage

```
te(mrs_data)
```

Arguments

mrs_data	MRS data.

Value

echo time in seconds.

`tr` *Return the repetition time of an MRS dataset.*

Description

Return the repetition time of an MRS dataset.

Usage

```
tr(mrs_data)
```

Arguments

`mrs_data`	MRS data.

Value

repetition time in seconds.

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

Description

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

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_basic_opts` *Return a list of options for a basic VARPRO analysis.*

Description

Return a list of options for a basic VARPRO analysis.

Usage

```
varpro_basic_opts(method = "fd_re", nnls = TRUE, ppm_left = 4, ppm_right = 0.2)
```

Arguments

`method`	one of "td", "fd", "fd_re".
`nnls`	restrict basis amplitudes to non-negative values.
`ppm_left`	downfield frequency limit for the fitting range (ppm).
`ppm_right`	upfield frequency limit for the fitting range (ppm).

Value

full 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(),
  nuc = def_nuc(),
  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.
nuc	resonant nucleus.
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(mrs_data, fname, format = NULL, force = FALSE)
```

Arguments

`mrs_data`	object to be written to file, or list of mrs_data objects.
`fname`	one or more filenames to output.
`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.
`force`	set to TRUE to overwrite any existing files.

`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(mrs_data, fname)
```

Arguments

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

`write_pulse_ascii` *Write an ASCII formatted pulse file.*

Description

Write an ASCII formatted pulse file.

Usage

```
write_pulse_ascii(pulse, path)
```

Arguments

`pulse`	pulse data object.
`path`	file path for export.

`zero_fade_spec` — *Fade a spectrum to zero by frequency domain multiplication with a tanh function. Note this operation distorts data points at the end of the FID.*

Description

Fade a spectrum to zero by frequency domain multiplication with a tanh function. Note this operation distorts data points at the end of the FID.

Usage

```
zero_fade_spec(mrs_data, start_ppm, end_ppm)
```

Arguments

`mrs_data`	data to be faded.
`start_ppm`	start point of the fade in ppm units.
`end_ppm`	end point of the fade in ppm units.

Value

modified mrs_data object.

`zero_higher_orders` — *Zero all coherences including and above a given order.*

Description

Zero all coherences including and above a given order.

Usage

```
zero_higher_orders(sys, rho, order)
```

Arguments

`sys`	spin system object.
`rho`	density matrix.
`order`	states higher than or equal to this argument will be set to zero.

Value

density matrix.

`zero_td_pts_end` *Set `mrs_data` object data points at the end of the FID to zero.*

Description

Set `mrs_data` object data points at the end of the FID to zero.

Usage

```
zero_td_pts_end(mrs_data, pts)
```

Arguments

`mrs_data`	MRS data.
`pts`	number of end points to set to zero.

Value

modified `mrs_data` object.

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

Description

Zero-fill MRS data in the time domain.

Usage

```
zf(x, factor = 2, offset = 0)

## S3 method for class 'list'
zf(x, factor = 2, offset = 0)

## S3 method for class 'mrs_data'
zf(x, factor = 2, offset = 0)

## S3 method for class 'basis_set'
zf(x, factor = 2, offset = 0)
```

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.
`offset`	number of points from the end of the FID to insert the zero values.

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, a factor of 2 returns a dataset with twice the original points in the x-y directions. Factors smaller than one are permitted, such that a factor of 0.5 returns half the k-space points in the x-y directions.

Value

zero-filled data.

Index