

# Package ‘spant’

May 28, 2019

**Type** Package

**Title** MR Spectroscopy Analysis Tools

**Version** 0.14.0

**Date** 2019-05-28

**Description** Tools for reading, visualising and processing Magnetic Resonance Spectroscopy data.

**BugReports** <https://github.com/martin3141/spant/issues>

**License** GPL-3

**RoxygenNote** 6.1.1

**LazyData** yes

**Depends** R (>= 2.10)

**Imports** abind, plyr, foreach, pracma, stringr, complexplus, signal, matrixcalc, minpack.lm, nls, utils, graphics, grDevices, smoother, svd, tkrplot, readr, magrittr, tibble

**Suggests** oro.nifti, MASS, fastICA, fields, mmand, ptw, viridisLite, viridis, RNifti, RNiftyReg, neurobase, knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**Encoding** UTF-8

**NeedsCompilation** no

**Author** Martin Wilson [cre, aut],  
John Muschelli [ctb]

**Maintainer** Martin Wilson <[martin@pipegrep.co.uk](mailto:martin@pipegrep.co.uk)>

**Repository** CRAN

**Date/Publication** 2019-05-28 15:30:04 UTC

**R topics documented:**

spant-package . . . . .	6
acquire . . . . .	6
align . . . . .	7
apodise_xy . . . . .	7
append_dyns . . . . .	8
apply_axes . . . . .	8
apply_mrs . . . . .	9
apply_pvc . . . . .	9
Arg.mrs_data . . . . .	10
auto_phase . . . . .	10
basis2mrs_data . . . . .	11
beta2lw . . . . .	11
calc_coil_noise_cor . . . . .	12
calc_coil_noise_sd . . . . .	12
calc_sd_poly . . . . .	13
calc_spec_diff . . . . .	13
calc_spec_snr . . . . .	14
check_lcm . . . . .	14
check_tqn . . . . .	15
collapse_to_dyns . . . . .	15
comb_coils . . . . .	15
comb_csv_results . . . . .	16
comb_metab_ref . . . . .	17
conj . . . . .	17
Conj.mrs_data . . . . .	18
conv_mrs . . . . .	18
crop_spec . . . . .	19
crop_td_pts . . . . .	19
decimate_mrs . . . . .	20
def_acq_paras . . . . .	20
def_fs . . . . .	21
def_ft . . . . .	21
def_N . . . . .	22
def_ref . . . . .	22
diff_mrs . . . . .	22
dyns . . . . .	23
ecc . . . . .	23
est_noise_sd . . . . .	24
fd2td . . . . .	24
fd_conv_filt . . . . .	25
fit_amps . . . . .	25
fit_diags . . . . .	26
fit_mrs . . . . .	26
fit_tab2csv . . . . .	27
fp_phase . . . . .	28
fp_phase_correct . . . . .	28

fs	29
ft_shift	29
ft_shift_mat	30
gen_F	30
gen_F_xy	31
get_1h_brain_basis_paras	31
get_1h_brain_basis_paras_v1	32
get_1h_brain_basis_paras_v2	32
get_acq_paras	33
get_dyns	33
get_even_dyns	34
get_fh_dyns	34
get_fit_map	35
get_fp	35
get_guassain_pulse	36
get_metab	36
get_mol_names	37
get_mol_paras	37
get_odd_dyns	38
get_ref	38
get_seg_ind	39
get_sh_dyns	39
get_slice	40
get_subset	40
get_svs_voi	41
get_td_amp	41
get_uncoupled_mol	42
get_voi_seg	42
get_voxel	43
hsvd_filt	43
ift_shift	44
Im.mrs_data	44
image.mrs_data	45
interleave_dyns	46
int_spec	46
inv_even_dyns	47
inv_odd_dyns	47
is_fd	48
lb	48
lw2alpha	49
lw2beta	49
mat2mrs_data	50
max_mrs	50
max_mrs_interp	51
mean.mrs_data	51
mean_dyns	52
mean_dyn_blocks	52
mean_dyn_pairs	53

median_dyns . . . . .	53
Mod.mrs_data . . . . .	54
mrs_data2basis . . . . .	54
mrs_data2mat . . . . .	55
mrs_data2vec . . . . .	55
mvfftshift . . . . .	56
N . . . . .	56
n2coord . . . . .	57
Nspec . . . . .	57
peak_info . . . . .	57
phase . . . . .	58
plot.fit_result . . . . .	59
plot.mrs_data . . . . .	60
plot_fit_slice . . . . .	61
plot_fit_slice_inter . . . . .	61
plot_slice_map . . . . .	62
plot_slice_map_inter . . . . .	62
plot_voi_overlay . . . . .	63
plot_voi_overlay_seg . . . . .	63
ppm . . . . .	64
print.fit_result . . . . .	64
print.mrs_data . . . . .	65
qn_states . . . . .	65
rats . . . . .	66
Re.mrs_data . . . . .	66
read_basis . . . . .	67
read_lcm_coord . . . . .	67
read_mrs . . . . .	68
read_mrs_dpt . . . . .	69
read_mrs_tqn . . . . .	69
read_tqn_fit . . . . .	70
read_tqn_result . . . . .	71
rep_array_dim . . . . .	71
rep_dyn . . . . .	72
rep_mrs . . . . .	72
resample_voi . . . . .	73
rm_dyns . . . . .	73
scale_amp_molal_pvc . . . . .	74
scale_amp_molar . . . . .	74
scale_amp_ratio . . . . .	75
scale_amp_water_ratio . . . . .	75
seconds . . . . .	76
seq_cpmg_ideal . . . . .	76
seq_mega_press_ideal . . . . .	77
seq_press_ideal . . . . .	77
seq_pulse_acquire . . . . .	78
seq_pulse_acquire_31p . . . . .	79
seq_slaser_ideal . . . . .	79

seq_spin_echo_ideal . . . . .	80
seq_spin_echo_ideal_31p . . . . .	80
seq_steam_ideal . . . . .	81
set_def_acq_paras . . . . .	81
set_lcm_cmd . . . . .	82
set_ref . . . . .	82
set_td_pts . . . . .	82
set_tqn_cmd . . . . .	83
shift . . . . .	83
sim_basis . . . . .	84
sim_basis_1h_brain . . . . .	84
sim_basis_1h_brain_press . . . . .	85
sim_basis_tqn . . . . .	85
sim_brain_1h . . . . .	86
sim_mol . . . . .	87
sim_noise . . . . .	87
sim_resonances . . . . .	88
spant_mpress_drift . . . . .	89
spin_sys . . . . .	89
spm_pve2categorical . . . . .	90
stackplot . . . . .	90
stackplot.fit_result . . . . .	91
stackplot.mrs_data . . . . .	92
sum_coils . . . . .	93
sum_dyns . . . . .	93
td2fd . . . . .	94
tdsr . . . . .	94
td_conv_filt . . . . .	95
varpro_3_para_opts . . . . .	95
varpro_opts . . . . .	96
vec2mrs_data . . . . .	97
write_basis . . . . .	97
write_basis_tqn . . . . .	98
write_mrs_dpt_v2 . . . . .	98
write_mrs_lcm_raw . . . . .	99
zero_nzoc . . . . .	99
zf . . . . .	100
zf_xy . . . . .	100

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 learn more about spant, start with the vignettes: `browseVignettes(package = "spant")`

For a full list of functions: `help(package=spant,help_type="html")`

### Author(s)

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

Other contributors:

- John Muschelli [contributor]

### See Also

Useful links:

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

acquire                    *Simulate pulse sequence acquisition.*

---

### Description

Simulate pulse sequence acquisition.

### Usage

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

### Arguments

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

### Value

a list of resonance amplitudes and frequencies.

---

align	<i>Align spectra to a reference frequency using a convolution based method.</i>
-------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

aligned data object.

---

apodise_xy	<i>Apodise MRSI data in the x-y direction with a k-space hamming filter.</i>
------------	--

---

**Description**

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

**Usage**

```
apodise_xy(mrs_data)
```

**Arguments**

mrs_data	MRSI data.
----------	------------

**Value**

apodised data.

---

append_dyns	<i>Append MRS data across the dynamic dimension, assumes they matched across the other dimensions.</i>
-------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

---

apply_axes	<i>Apply a function over specified array axes.</i>
------------	--

---

**Description**

Apply a function over specified array axes.

**Usage**

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

**Arguments**

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

**Value**

array.



**Examples**

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

---

 apply\_mrs

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


---

**Description**

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

**Usage**

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

**Arguments**

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

---

apply\_pvc

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


---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

a `fit_result` object with an added results table: "res\_tab\_molal\_pvc".

---

Arg.mrs_data	<i>Apply Arg operator to an MRS dataset.</i>
--------------	--

---

**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.

---

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

---

**Description**

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

**Usage**

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

**Arguments**

mrs\_data            an object of class `mrs_data`.  
 xlim                frequency range (default units of PPM) to including in the phase  
 ret\_phase          return phase values (logical). determination.

**Value**

MRS data object and phase values (optional).

---

basis2mrs_data	<i>Convert a basis object to an mrs_data object - where basis signals are spread across the dynamic dimension.</i>
----------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

an mrs\_data object with basis signals spread across the dynamic dimension or summed.

---

beta2lw	<i>Covert a beta value in the time-domain to an equivalent linewidth in Hz: <math>x * \exp(-i * t * t * \text{beta})</math>.</i>
---------	--

---

**Description**

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

**Usage**

```
beta2lw(beta)
```

**Arguments**

beta	beta damping value.
------	---------------------

**Value**

linewidth value in Hz.

calc\_coil\_noise\_cor     *Calculate the noise correlation between coil elements.*

---

**Description**

Calculate the noise correlation between coil elements.

**Usage**

```
calc_coil_noise_cor(noise_data)
```

**Arguments**

noise\_data     mrs\_data object with one FID for each coil element.

**Value**

correlation matrix.

---

calc\_coil\_noise\_sd     *Calculate the noise standard deviation for each coil element.*

---

**Description**

Calculate the noise standard deviation for each coil element.

**Usage**

```
calc_coil_noise_sd(noise_data)
```

**Arguments**

noise\_data     mrs\_data object with one FID for each coil element.

**Value**

array of standard deviations.

---

calc_sd_poly	<i>Perform a polynomial fit, subtract and return the standard deviation of the residuals.</i>
--------------	---

---

**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	<i>Calculate the sum of squares differences between two mrs_data objects.</i>
----------------	---

---

**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	<i>Calculate the spectral SNR.</i>
---------------	------------------------------------

---

**Description**

SNR is defined as the maximum signal value divided by 2 times 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 separatly.

**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	<i>Check LCModel can be run</i>
-----------	---------------------------------

---

**Description**

Check LCModel can be run

**Usage**

```
check_lcm()
```

---

check_tqn	<i>Check the TARQUIN binary can be run</i>
-----------	--

---

**Description**

Check the TARQUIN binary can be run

**Usage**

check\_tqn()

---

collapse_to_dyns	<i>Collapse MRS data by concatenating spectra along the dynamic dimension.</i>
------------------	--

---

**Description**

Collapse MRS data by concatenating spectra along the dynamic dimension.

**Usage**

collapse\_to\_dyns(mrs\_data)

**Arguments**

mrs\_data      mrs\_data object to be collapsed.

**Value**

collapsed data with spectra concatenated along the dynamic dimension.

---

comb_coils	<i>Combine coil data based on the first data point of a reference signal.</i>
------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

MRS data.

---

comb_csv_results	<i>Combine the results from multiple csv format files into a table.</i>
------------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

results table.



---

comb_metab_ref	<i>Combine a reference and metabolite mrs_data object.</i>
----------------	--

---

**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	<i>Conjugate MRS data.</i>
------	----------------------------

---

**Description**

Conjugate MRS data.

**Usage**

```
conj(mrs_data)
```

**Arguments**

mrs_data	input data.
----------	-------------

**Value**

conjugated data.

---

Conj.mrs_data	<i>Apply Conj operator to an MRS dataset.</i>
---------------	---

---

**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	<i>Convolve two MRS data objects.</i>
----------	---------------------------------------

---

**Description**

Convolve two MRS data objects.

**Usage**

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

**Arguments**

mrs\_data           MRS data to be convolved.  
conv               convolution data stored as an mrs\_data object.

**Value**

convolved data.

---

crop_spec	<i>Crop mrs_data object based on a frequency range.</i>
-----------	---

---

**Description**

Crop mrs\_data object based on a frequency range.

**Usage**

```
crop_spec(mrs_data, xlim = c(4, 0.5), scale = "ppm")
```

**Arguments**

mrs_data	MRS data.
xlim	range of values to crop in the spectral dimension eg xlim = c(4,0.5).
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	<i>Crop mrs_data object data points in the time-domain.</i>
-------------	---

---

**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.

---

decimate_mrs	<i>Downsample an MRS signal by a factor.</i>
--------------	--

---

**Description**

Downsample an MRS signal by a factor.

**Usage**

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

**Arguments**

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

**Value**

downsampled data.

---

def_acq_paras	<i>Return (and optionally modify using the input arguments) a list of the default acquisition parameters.</i>
---------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

A list containing the following elements:

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

---

def_fs	<i>Return the default sampling frequency in Hz.</i>
--------	---

---

**Description**

Return the default sampling frequency in Hz.

**Usage**

```
def_fs()
```

**Value**

sampling frequency in Hz.

---

def_ft	<i>Return the default transmitter frequency in Hz.</i>
--------	--

---

**Description**

Return the default transmitter frequency in Hz.

**Usage**

```
def_ft()
```

**Value**

transmitter frequency in Hz.

---

def_N	<i>Return the default number of data points in the spectral dimension.</i>
-------	--

---

**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_ref	<i>Return the default reference value for ppm scale.</i>
---------	--

---

**Description**

Return the default reference value for ppm scale.

**Usage**

def\_ref()

**Value**

reference value for ppm scale.

---

diff_mrs	<i>Apply the diff operator to an MRS dataset in the FID/spectral dimension.</i>
----------	---

---

**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.

---

dys                            *Return the number of dynamic scans in an MRS dataset.*

---

**Description**

Return the number of dynamic scans in an MRS dataset.

**Usage**

dys(mrs\_data)

**Arguments**

mrs\_data        MRS data.

**Value**

number of dynamic scans.

---

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              ceference dataset.  
 rev              reverse the correction.

**Details**

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

**Value**

corrected data in the time domain.

---

est_noise_sd	<i>Estimate the standard deviation of the noise from a segment of an mrs_data object.</i>
--------------	---

---

**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	<i>Transform frequency-domain data to the time-domain.</i>
-------	--

---

**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	<i>Frequency-domain convolution based filter.</i>
--------------	---

---

**Description**

Frequency-domain convolution based filter.

**Usage**

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

**Arguments**

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

---

fit_amps	<i>Extract the fit amplitues from an object of class fit_result.</i>
----------	--

---

**Description**

Extract the fit amplitues from an object of class fit\_result.

**Usage**

```
fit_amps(x, inc_index = FALSE, sort_names = TRUE,
         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	<i>Calculate diagnostic information for object of class fit_result.</i>
-----------	---

---

**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	<i>Perform a fit based analysis of MRS data.</i>
---------	--

---

**Description**

Note that TARQUIN and LCModel require these packages to be installed, and the functions set\_tqn\_path and set\_lcm\_path (respectively) need to be used to specify the location of these software packages.

**Usage**

```
fit_mrs(metab, basis = NULL, method = "VARPRO_3P", w_ref = NULL,
        opts = NULL, parallel = FALSE, cores = 4)
```

**Arguments**

metab	metabolite data.
basis	basis class object or character vector to basis file in LCModel .basis format.
method	'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 analysis in parallel (TRUE or FALSE).
cores	number of cores to use for parallel analysis.

## Details

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

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

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

## Value

MRS analysis object.

## Examples

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

## End(Not run)
```

---

fit\_tab2csv

*Write fit results table to a csv file.*

---

## Description

Write fit results table to a csv file.

## Usage

```
fit_tab2csv(x, fname, pvc = FALSE)
```

## Arguments

x	fit results table.
fname	filename of csv file.
pvc	output PVC or raw results (logical).

---

fp_phase	<i>Return the phase of the first data point in the time-domain.</i>
----------	---

---

**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	<i>Perform a zeroth order phase correction based on the phase of the first data point in the time-domain.</i>
------------------	---

---

**Description**

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

**Usage**

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

**Arguments**

mrs\_data      MRS data to be corrected.  
ret\_phase      return phase values (logical).

**Value**

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

---

fs	<i>Return the sampling frequency in Hz of an MRS dataset.</i>
----	---

---

**Description**

Return the sampling frequency in Hz of an MRS dataset.

**Usage**

```
fs(mrs_data)
```

**Arguments**

mrs\_data      MRS data.

**Value**

sampling frequency in Hz.

---

ft_shift	<i>Perform a fft and fshift on a vector.</i>
----------	--

---

**Description**

Perform a fft and fshift on a vector.

**Usage**

```
ft_shift(vec_in)
```

**Arguments**

vec\_in      vector input.

**Value**

output vector.

---

ft_shift_mat	<i>Perform a fft and fftshift on a matrix with each column replaced by its shifted fft.</i>
--------------	---

---

**Description**

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

**Usage**

```
ft_shift_mat(mat_in)
```

**Arguments**

mat_in	matrix input.
--------	---------------

**Value**

output matrix.

---

gen_F	<i>Generate the F product operator.</i>
-------	---

---

**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	<i>Generate the Fxy product operator with a specified phase.</i>
----------	--

---

**Description**

Generate the Fxy product operator with a specified phase.

**Usage**

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

**Arguments**

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

**Value**

product operator matrix.

---

get_1h_brain_basis_paras	<i>Return a list of mol_parameter objects suitable for 1H brain MRS analyses.</i>
--------------------------	---

---

**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_acq_paras	<i>Return acquisition parameters from a MRS data object.</i>
---------------	--

---

**Description**

Return acquisition parameters from a MRS data object.

**Usage**

```
get_acq_paras(mrs_data)
```

**Arguments**

mrs_data	MRS data.
----------	-----------

**Value**

list of acquisition parameters.

---

get_dyns	<i>Extract a subset of dynamic scans.</i>
----------	---

---

**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	<i>Return even numbered dynamic scans starting from 1 (2,4,6...).</i>
---------------	---

---

**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	<i>Return the first half of a dynamic series.</i>
-------------	---

---

**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	<i>Get a data array from a fit result.</i>
-------------	--

---

**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	<i>Return the first time-domain data point.</i>
--------	---

---

**Description**

Return the first time-domain data point.

**Usage**

```
get_fp(mrs_data)
```

**Arguments**

mrs_data	MRS data.
----------	-----------

**Value**

first time-domain data point.

---

get\_guassian\_pulse      *Generate a gaussian pulse shape.*

---

**Description**

Generate a gaussian pulse shape.

**Usage**

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

**Arguments**

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

---

get\_metab      *Extract the metabolite component from an mrs\_data object.*

---

**Description**

Extract the metabolite component from an mrs\_data object.

**Usage**

```
get_metab(mrs_data)
```

**Arguments**

mrs_data	MRS data.
----------	-----------

**Value**

metabolite component.

---

get_mol_names	<i>Return a character array of names that may be used with the get_mol_paras function.</i>
---------------	--

---

**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	<i>Get a mol_parameters object for a named molecule.</i>
---------------	--

---

**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_odd_dyns	<i>Return odd numbered dynamic scans starting from 1 (1,3,5...).</i>
--------------	--

---

**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	<i>Extract the reference component from an mrs_data object.</i>
---------	---

---

**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	<i>Get the indices of data points lying between two values (end &gt; x &gt; start).</i>
-------------	---

---

**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	<i>Return the second half of a dynamic series.</i>
-------------	--

---

**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	<i>Return a single slice from a larger MRSI dataset.</i>
-----------	--

---

**Description**

Return a single slice from a larger MRSI dataset.

**Usage**

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

**Arguments**

mrs_data	MRSI data.
z_pos	the z index to extract.

**Value**

MRS data.

---

get_subset	<i>Extract a subset of MRS data.</i>
------------	--------------------------------------

---

**Description**

Extract a subset of MRS data.

**Usage**

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

**Arguments**

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

**Value**

selected subset of MRS data.



---

get_svs_voi	<i>Generate a SVS acquisition volume from an mrs_data object.</i>
-------------	---

---

**Description**

Generate a SVS acquisition volume from an mrs\_data object.

**Usage**

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

**Arguments**

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

**Value**

volume data as a nifti object.

---

get_td_amp	<i>Return an array of amplitudes derived from fitting the initial points in the time domain and extrapolating back to t=0.</i>
------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

array of amplitudes.

---

get_uncoupled_mol	<i>Generate a mol_parameters object for a simple spin system with one resonance.</i>
-------------------	--

---

**Description**

Generate a mol\_parameters object for a simple spin system with one resonance.

**Usage**

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

**Arguments**

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

**Value**

mol\_parameters object.

---

get_voi_seg	<i>Return the white matter, gray matter and CSF composition of a volume.</i>
-------------	--

---

**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_voxel	<i>Return a single voxel from a larger mrs dataset.</i>
-----------	---

---

**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.

---

hsvd_filt	<i>HSVD based signal filter.</i>
-----------	----------------------------------

---

**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 = 50, propack = FALSE)
```

**Arguments**

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

---

ift_shift	<i>Perform an iffshift and ifft on a vector.</i>
-----------	--

---

**Description**

Perform an iffshift and ifft on a vector.

**Usage**

```
ift_shift(vec_in)
```

**Arguments**

vec\_in            vector input.

**Value**

output vector.

---

Im.mrs_data	<i>Apply Im operator to an MRS dataset.</i>
-------------	---

---

**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	<i>Image plot method for objects of class mrs_data.</i>
----------------	---

---

## Description

Image plot method for objects of class mrs\_data.

## Usage

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

## Arguments

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

---

interleave_dyns	<i>Interleave the first and second half of a dynamic series.</i>
-----------------	--

---

**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	<i>Integrate a spectral region.</i>
----------	-------------------------------------

---

**Description**

Integrate a spectral region.

**Usage**

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

**Arguments**

mrs_data	MRS data.
xlim	spectral range to be integrated.
scale	units of xlim, can be : "ppm", "Hz" or "points".
mode	spectral mode, can be : "re", "im" or "mod".

**Value**

an array of integral values.

---

inv_even_dyns	<i>Invert even numbered dynamic scans starting from 1 (2,4,6...).</i>
---------------	---

---

**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	<i>Invert odd numbered dynamic scans starting from 1 (1,3,5...).</i>
--------------	--

---

**Description**

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

**Usage**

```
inv_odd_dyns(mrs_data)
```

**Arguments**

mrs\_data      dynamic MRS data.

**Value**

dynamic MRS data with inverted odd numbered scans.

---

is_fd	<i>Check if the chemical shift dimension of an MRS data object is in the frequency domain.</i>
-------	--

---

**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.

---

lb	<i>Apply line-broadening (apodisation) to MRS data or basis object.</i>
----	---

---

**Description**

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

**Usage**

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

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

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

**Arguments**

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

**Value**

line-broadened data.



---

lw2alpha	<i>Covert a linewidth in Hz to an equivalent alpha value in the time-domain ie: <math>x * \exp(-t * \alpha)</math>.</i>
----------	---

---

**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	<i>Covert a linewidth in Hz to an equivalent beta value in the time-domain ie: <math>x * \exp(-t * t * \beta)</math>.</i>
---------	---

---

**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.

---

mat2mrs_data	<i>Convert a matrix (with spectral points in the row dimension and dynamics in the column dimensions) into a mrs_data object.</i>
--------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

mrs\_data object.

---

max_mrs	<i>Apply the max operator to an MRS dataset.</i>
---------	--

---

**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	<i>Apply the max operator to an interpolated MRS dataset.</i>
----------------	---

---

**Description**

Apply the max operator to an interpolated MRS dataset.

**Usage**

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

**Arguments**

mrs_data	MRS data.
interp_f	interpolation factor.

**Value**

Array of maximum values (real only).

---

mean.mrs_data	<i>Calculate the mean spectrum from an mrs_data object.</i>
---------------	---

---

**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	<i>Calculate the mean dynamic data.</i>
-----------	---

---

**Description**

Calculate the mean dynamic data.

**Usage**

```
mean_dyns(mrs_data)
```

**Arguments**

mrs_data	dynamic MRS data.
----------	-------------------

**Value**

mean dynamic data.

---

mean_dyn_blocks	<i>Calculate the mean of adjacent dynamic scans.</i>
-----------------	--

---

**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	<i>Calculate the pairwise means across a dynamic data set.</i>
----------------	--

---

**Description**

Calculate the pairwise means across a dynamic data set.

**Usage**

```
mean_dyn_pairs(mrs_data)
```

**Arguments**

mrs\_data      dynamic MRS data.

**Value**

mean dynamic data of adjacent dynamic pairs.

---

median_dyns	<i>Calculate the median dynamic data.</i>
-------------	---

---

**Description**

Calculate the median dynamic data.

**Usage**

```
median_dyns(mrs_data)
```

**Arguments**

mrs\_data      dynamic MRS data.

**Value**

median dynamic data.

---

Mod.mrs_data	<i>Apply Mod operator to an MRS dataset.</i>
--------------	--

---

**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.

---

mrs_data2basis	<i>Convert an mrs_data object to basis object - where basis signals are spread across the dynamic dimension in the MRS data.</i>
----------------	--

---

**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	<i>Convert mrs_data object to a matrix, with spectral points in the row dimension and dynamics in the column dimension.</i>
--------------	---

---

**Description**

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

**Usage**

```
mrs_data2mat(mrs_data)
```

**Arguments**

mrs_data	MRS data object.
----------	------------------

**Value**

MRS data matrix.

---

mrs_data2vec	<i>Convert mrs_data object to a vector.</i>
--------------	---

---

**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.

---

<code>mvfftshift</code>	<i>Perform a fftshift on a matrix, with each column replaced by its shifted result.</i>
-------------------------	---

---

**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.

---

<code>N</code>	<i>Return the number of data points in an MRS dataset.</i>
----------------	--

---

**Description**

Return the number of data points in an MRS dataset.

**Usage**

```
N(mrs_data)
```

**Arguments**

`mrs_data` MRS data.

**Value**

number of data points.



---

n2coord	<i>Print fit coordinates from a single index.</i>
---------	---

---

**Description**

Print fit coordinates from a single index.

**Usage**

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

**Arguments**

n	fit index.
fit_res	fit_result object.

---

Nspec	<i>Return the total number of spectra in an MRS dataset.</i>
-------	--

---

**Description**

Return the total number of spectra in an MRS dataset.

**Usage**

```
Nspec(mrs_data)
```

**Arguments**

mrs_data	MRS data.
----------	-----------

---

peak_info	<i>Search for the highest peak in a spectral region and return the frequency, height and FWHM.</i>
-----------	--

---

**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**

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

---

<code>phase</code>	<i>Apply phasing parameters to MRS data.</i>
--------------------	--

---

**Description**

Apply phasing parameters to MRS data.

**Usage**

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

**Arguments**

<code>mrs_data</code>	MRS data.
<code>zero_order</code>	zero'th order phase term in degrees.
<code>first_order</code>	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, xlim = NULL, data_only = FALSE,
     label = NULL, plot_sigs = NULL, dyn = 1, x_pos = 1, y_pos = 1,
     z_pos = 1, coil = 1, n = NULL, sub_bl = FALSE, mar = NULL,
     restore_def_par = TRUE, ylim = NULL, y_scale = FALSE, ...)
```

### Arguments

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

---

plot.mrs\_data

*Plotting method for objects of class mrs\_data.*


---

## Description

Plotting method for objects of class `mrs_data`.

## Usage

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

## Arguments

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

---

plot\_fit\_slice      *Plot a 2D slice from an MRSI fit result object.*

---

**Description**

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

**Usage**

```
plot_fit_slice(fit_res, name, slice = 1, zlim = NULL, interp = 1)
```

**Arguments**

fit_res	fit_result object.
name	name of the quantity to plot, eg "TNAA".
slice	slice to plot in the z direction.
zlim	range of values to plot.
interp	interpolation factor.

---

plot\_fit\_slice\_inter      *Plot a 2D slice from an MRSI fit result object.*

---

**Description**

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

**Usage**

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

**Arguments**

fit_res	fit_result object.
map	array of values to be plotted, defaults to a "TNAA" map.
slice	slice to plot in the z direction.
zlim	range of values to plot.
interp	interpolation factor.

---

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 colorbar 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)
```

**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.

---

plot\_voi\_overlay      *Plot a volume as an image overlay.*

---

**Description**

Plot a volume as an image overlay.

**Usage**

```
plot_voi_overlay(voi, mri)
```

**Arguments**

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

---

plot\_voi\_overlay\_seg      *Plot a volume as an overlay on a segmented brain volume.*

---

**Description**

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

**Usage**

```
plot_voi_overlay_seg(voi, mri_seg)
```

**Arguments**

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

---

ppm	<i>Return the ppm scale of an MRS dataset.</i>
-----	--

---

**Description**

Return the ppm scale of an MRS dataset.

**Usage**

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

**Arguments**

mrs_data	MRS data.
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.

**Value**

ppm scale.

---

print.fit_result	<i>Print a summary of an object of class fit_result.</i>
------------------	--

---

**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	<i>Print a summary of mrs_data parameters.</i>
----------------	--

---

**Description**

Print a summary of mrs\_data parameters.

**Usage**

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

**Arguments**

x	mrs_data object.
...	further arguments.

---

qn_states	<i>Get the quantum coherence matrix for a spin system.</i>
-----------	--

---

**Description**

Get the quantum coherence matrix for a spin system.

**Usage**

```
qn_states(sys)
```

**Arguments**

sys	spin system object.
-----	---------------------

**Value**

quantum coherence number matrix.

---

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

---

### Description

Robust Alignment to a Target Spectrum (RATS).

### Usage

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

### Arguments

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

### Value

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

---

Re.mrs\_data *Apply Re operator to an MRS dataset.*

---

### Description

Apply Re operator to an MRS dataset.

### Usage

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

### Arguments

z	MRS data.
---	-----------

### Value

MRS data following Re operator.

---

read_basis	<i>Read a basis file in LCMModel .basis format.</i>
------------	---

---

**Description**

Read a basis file in LCMModel .basis format.

**Usage**

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

**Arguments**

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

**Value**

basis object.

---

read_lcm_coord	<i>Read an LCMModel formatted coord file containing fit information.</i>
----------------	--

---

**Description**

Read an LCMModel 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	<i>Read MRS data from a file.</i>
----------	-----------------------------------

---

### Description

Read MRS data from a file.

### Usage

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

### Arguments

fname	filename of the dpt format MRS data.
format	string describing the data format. May be one of the following : "spar_sdat", "rda", "ima", "twix", "pfile", "list_data", "paravis", "dpt".
ft	transmitter frequency in Hz (required for list_data format).
fs	sampling frequency in Hz (required for list_data format).
ref	reference value for ppm scale (required for list_data format).
n_ref_scans	override the number of water reference scans detected in the file header (GE p-file only).
full_data	export all data points, including those before the start of the FID (default = FALSE).
verbose	print data file information (default = FALSE).

### Value

MRS data object.

### Examples

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

---

read_mrs_dpt	<i>Read MRS data stored in dangerplot (dpt) v3 format.</i>
--------------	--

---

**Description**

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

**Usage**

```
read_mrs_dpt(fname)
```

**Arguments**

fname            filename of the dpt format MRS data.

**Value**

MRS data object.

**Examples**

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

## End(Not run)
```

---

read_mrs_tqn	<i>Read MRS data using the TARQUIN software package.</i>
--------------	--

---

**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\_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	<i>Reader for csv results generated by TARQUIN.</i>
-----------------	---

---

**Description**

Reader for csv results generated by TARQUIN.

**Usage**

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

**Arguments**

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

**Value**

list of amplitudes, crlbs and diagnostics.

**Examples**

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

---

rep_array_dim	<i>Repeat an array over a given dimension.</i>
---------------	--

---

**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	<i>Replicate a scan in the dynamic dimension.</i>
---------	---

---

**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	<i>Replicate a scan over a given dimension.</i>
---------	---

---

**Description**

Replicate a scan over a given dimension.

**Usage**

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

**Arguments**

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

**Value**

replicated data object.



---

resample_voi	<i>Resample a VOI to match a target image space.</i>
--------------	--

---

**Description**

Resample a VOI to match a target image space.

**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.

---

rm_dyns	<i>Remove a subset of dynamic scans.</i>
---------	--

---

**Description**

Remove a subset of dynamic scans.

**Usage**

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

**Arguments**

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

**Value**

MRS data without the specified dynamic scans.

---

scale\_amp\_molal\_pvc     *Apply partial volume correction to a fitting result object.*

---

### Description

Apply partial volume correction to a fitting result object.

### Usage

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

### Arguments

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

### Value

A fit\_result object with a res\_tab\_molal\_pvc data table added.

---

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

---

### Description

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

### Usage

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

### Arguments

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

**Value**

a `fit_result` object with a `res_tab_molar` data table added.

---

scale_amp_ratio	<i>Scale fitted amplitudes to a ratio of signal amplitude.</i>
-----------------	--

---

**Description**

Scale fitted amplitudes to a ratio of signal amplitude.

**Usage**

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

**Arguments**

<code>fit_result</code>	a result object generated from fitting.
<code>name</code>	the signal name to use as a denominator (usually, "TCr" or "TNAA").

**Value**

a `fit_result` object with a `res_tab_ratio` data table added.

---

scale_amp_water_ratio	<i>Scale metabolite amplitudes as a ratio to the unsupressed water amplitude.</i>
-----------------------	---

---

**Description**

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

**Usage**

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

**Arguments**

<code>fit_result</code>	a result object generated from fitting.
<code>ref_data</code>	a water reference MRS data object.

**Value**

a `fit_result` object with a `res_tab_water_ratio` data table added.

---

seconds	<i>Return a time scale vector to match the FID of an MRS data object.</i>
---------	---

---

**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	<i>CPMG style sequence with ideal pulses.</i>
----------------	---

---

**Description**

CPMG style sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

seq\_mega\_press\_ideal    *MEGA-PRESS sequence with ideal localisation pulses and Gaussian shaped editing pulse.*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

seq\_press\_ideal    *PRESS sequence with ideal pulses.*

---

**Description**

PRESS sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

seq_pulse_acquire	<i>Simple pulse and acquire sequence with ideal pulses.</i>
-------------------	---

---

**Description**

Simple pulse and acquire sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

seq\_pulse\_acquire\_31p *Simple pulse and acquire sequence with ideal pulses.*

---

**Description**

Simple pulse and acquire sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

seq\_slaser\_ideal *sLASER sequence with ideal pulses.*

---

**Description**

sLASER sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

seq\_spin\_echo\_ideal    *Spin echo sequence with ideal pulses.*

---

**Description**

Spin echo sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

seq\_spin\_echo\_ideal\_31p  
*Spin echo sequence with ideal pulses.*

---

**Description**

Spin echo sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.



---

seq_steam_ideal	<i>STEAM sequence with ideal pulses.</i>
-----------------	--

---

**Description**

STEAM sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

set_def_acq_paras	<i>Set the default acquisition parameters.</i>
-------------------	--

---

**Description**

Set the default acquisition parameters.

**Usage**

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

**Arguments**

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

---

set_lcm_cmd	<i>Set the command to run the LCModel command-line program.</i>
-------------	---

---

**Description**

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

**Usage**

```
set_lcm_cmd(cmd)
```

**Arguments**

cmd	oath to binary.
-----	-----------------

---

set_ref	<i>Set the ppm reference value (eg ppm value at 0Hz).</i>
---------	---

---

**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	<i>Set the number of time-domain data points, truncating or zero-filling as appropriate.</i>
------------	--

---

**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	<i>Set the command to run the TARQUIN command-line program.</i>
-------------	---

---

**Description**

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

**Usage**

```
set_tqn_cmd(cmd)
```

**Arguments**

cmd	path to binary.
-----	-----------------

---

shift	<i>Apply a frequency shift to MRS data.</i>
-------	---

---

**Description**

Apply a frequency shift to MRS data.

**Usage**

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

**Arguments**

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

**Value**

frequency shifted MRS data.

---

sim_basis	<i>Simulate a basis set object.</i>
-----------	-------------------------------------

---

**Description**

Simulate a basis set object.

**Usage**

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

**Arguments**

mol_list	list of mol_parameter objects.
pul_seq	pulse sequence function to use.
acq_paras	list of acquisition parameters or an mrs_data object. See <a href="#">def_acq_paras</a>
xlim	ppm range limiting signals to be simulated.
...	extra parameters to pass to the pulse sequence function.

**Value**

basis object.

---

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

---

**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 <a href="#">def_acq_paras</a> .
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_params = def_acq_params(), xlim = c(0.5,
4.2), lcm_compat = FALSE, TE1 = 0.01, TE2 = 0.02)
```

**Arguments**

acq_params	list of acquisition parameters or an mrs_data object. See <a href="#">def_acq_params</a>
xlim	range of frequencies to simulate in ppm.
lcm_compat	exclude lipid and MM signals for use with default LCModel options.
TE1	TE1 of PRESS sequence (TE = TE1 + TE2).
TE2	TE2 of PRESS sequence.

**Value**

basis object.

---

sim\_basis\_tqn

*Simulate a basis file using TARQUIN.*

---

**Description**

Simulate a basis file using TARQUIN.

**Usage**

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

**Arguments**

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

**Examples**

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

## End(Not run)
```

---

sim_brain_1h	<i>Simulate MRS data with a similar appearance to normal brain (by default).</i>
--------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

see full\_output option.

---

sim_mol	<i>Simulate a mol_parameter object.</i>
---------	---

---

**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	<i>Simulate a time-domain mrs_data object containing simulated Gaussian noise.</i>
-----------	--

---

**Description**

Simulate a time-domain mrs\_data object containing simulated Gaussian noise.

**Usage**

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

**Arguments**

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

**Value**

mrs\_data object.

---

sim_resonances	<i>Simulate a MRS data object containing a set of simulated resonances.</i>
----------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

MRS data object.

**Examples**

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



---

spant\_mpress\_drift      *Example MEGA-PRESS data with significant B0 drift.*

---

**Description**

Example MEGA-PRESS data with significant B0 drift.

**Usage**

spant\_mpress\_drift

**Format**

An object of class `mrs_data` of dimension 1 x 1 x 1 x 1 x 40 x 1 x 1024.

---

spin\_sys      *Create a spin system object for pulse sequence simulation.*

---

**Description**

Create a spin system object for pulse sequence simulation.

**Usage**

spin\_sys(spin\_params, ft, ref)

**Arguments**

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

**Value**

spin system object.

---

<code>spm_pve2categorical</code>	<i>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.</i>
----------------------------------	---

---

**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.

---

<code>stackplot</code>	<i>Produce a plot with multiple traces.</i>
------------------------	---

---

**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 = 1, 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, ...)
```

### 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".
...	further arguments to plot method.

---

stackplot.mrs\_data      *Stackplot plotting method for objects of class mrs\_data.*

---

## Description

Stackplot plotting method for objects of class mrs\_data.

## Usage

```
## S3 method for class 'mrs_data'
stackplot(x, xlim = NULL, mode = "re",
  x_units = NULL, fd = TRUE, col = NULL, x_offset = 0,
  y_offset = 5, dim = "dyn", x_pos = NULL, y_pos = NULL,
  z_pos = NULL, dyn = 1, coil = 1, bty = NULL, labels = NULL,
  lab_cex = 1, right_marg = NULL, restore_def_par = TRUE, ...)
```

## Arguments

x	object of class mrs_data.
xlim	the range of values to display on the x-axis, eg xlim = c(4,1).
mode	representation of the complex numbers to be plotted, can be one of: "re", "im", "mod" or "arg".
x_units	the units to use for the x-axis, can be one of: "ppm", "hz", "points" or "seconds".
fd	display data in the frequency-domain (default), or time-domain (logical).
col	set the colour of the line, eg col = rgb(1,0,0,0.5).
x_offset	separate plots in the x-axis direction by this value. Default value is 0.
y_offset	separate plots in the y-axis direction by this value.
dim	the dimension to stack in the y-axis direction, can be one of: "dyn", "x", "y", "z" or "coil".
x_pos	the x index to plot.
y_pos	the y index to plot.
z_pos	the z index to plot.
dyn	the dynamic index to plot.
coil	the coil element number to plot.
bty	option to draw a box around the plot. See ?par.
labels	add labels to each data item.
lab_cex	label size.
right_marg	change the size of the right plot margin.
restore_def_par	restore default plotting par values after the plot has been made.
...	other arguments to pass to the matplot method.

---

sum_coils	<i>Calculate the sum across receiver coil elements.</i>
-----------	---

---

**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	<i>Calculate the sum of data dynamics.</i>
----------	--

---

**Description**

Calculate the sum of data dynamics.

**Usage**

```
sum_dyns(mrs_data)
```

**Arguments**

mrs\_data      dynamic MRS data.

**Value**

sum of data dynamics.

---

td2fd	<i>Transform time-domain data to the frequency-domain.</i>
-------	--

---

**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	<i>Time-domain spectral registration.</i>
------	---

---

**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	<i>Time-domain convolution based filter.</i>
--------------	--

---

### Description

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

### Usage

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

### Arguments

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

---

varpro_3_para_opts	<i>Return a list of options for VARPRO based fitting with 3 free parameters:</i>
	<ul style="list-style-type: none"> <li>• <i>zero'th order phase correction</i></li> <li>• <i>global damping</i></li> <li>• <i>global frequency shift.</i></li> </ul>

---

### Description

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

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

### Usage

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

**Arguments**

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

**Value**

list of options.

---

varpro_opts	<i>Return a list of options for VARPRO based fitting.</i>
-------------	---

---

**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	<i>Convert a vector into a mrs_data object.</i>
--------------	---

---

**Description**

Convert a vector into a mrs\_data object.

**Usage**

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

**Arguments**

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

**Value**

mrs\_data object.

---

write_basis	<i>Write a basis object to an LCModel .basis formatted file.</i>
-------------	--

---

**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\_dpt\_v2      *Write MRS data object to file in dangerplot (dpt) v2 format.*

---

### Description

Write MRS data object to file in dangerplot (dpt) v2 format.

### Usage

```
write_mrs_dpt_v2(fname, mrs_data)
```

### Arguments

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

### Examples

```
## Not run:
mrs_data <- write_mrs_dpt_v2("my_mrs_data.dpt", my_mrs_data)

## End(Not run)
```

---

write_mrs_lcm_raw	<i>Write MRS data object to file in a RAW format compatible with LCModel.</i>
-------------------	---

---

**Description**

Write MRS data object to file in a RAW format compatible with LCModel.

**Usage**

```
write_mrs_lcm_raw(fname, mrs_data, id = NA)
```

**Arguments**

fname	the filename of the output RAW format MRS data.
mrs_data	object to be written to file.
id	text string to identify the data.

**Examples**

```
## Not run:  
mrs_data <- write_mrs_lcm_raw("my_mrs_data.RAW", my_mrs_data)  
  
## End(Not run)
```

---

zero_nzoc	<i>Zero all non-zero-order coherences.</i>
-----------	--

---

**Description**

Zero all non-zero-order coherences.

**Usage**

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

**Arguments**

sys	spin system object.
rho	density matrix.

**Value**

density matrix.

---

zf	<i>Zero-fill MRS data in the time domain.</i>
----	---

---

**Description**

Zero-fill MRS data in the time domain.

**Usage**

```
zf(x, factor = 2)

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

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

**Arguments**

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

**Value**

zero-filled data.

---

zf_xy	<i>Zero-fill MRSI data in the k-space x-y direction.</i>
-------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

zero-filled data.

# Index

## \*Topic **datasets**

spant\_mpress\_drift, 89

acquire, 6

align, 7

apodise\_xy, 7

append\_dyns, 8

apply\_axes, 8

apply\_mrs, 9

apply\_pvc, 9

Arg.mrs\_data, 10

auto\_phase, 10

basis2mrs\_data, 11

beta2lw, 11

calc\_coil\_noise\_cor, 12

calc\_coil\_noise\_sd, 12

calc\_sd\_poly, 13

calc\_spec\_diff, 13

calc\_spec\_snr, 14

check\_lcm, 14

check\_tqn, 15

collapse\_to\_dyns, 15

comb\_coils, 15

comb\_csv\_results, 16

comb\_metab\_ref, 17

conj, 17

Conj.mrs\_data, 18

conv\_mrs, 18

crop\_spec, 19

crop\_td\_pts, 19

decimate\_mrs, 20

def\_acq\_params, 20, 84–86, 88

def\_fs, 21

def\_ft, 21

def\_N, 22

def\_ref, 22

diff\_mrs, 22

dyns, 23

ecc, 23

est\_noise\_sd, 24

fd2td, 24

fd\_conv\_filt, 25

fit\_amps, 25

fit\_diags, 26

fit\_mrs, 26

fit\_tab2csv, 27

fp\_phase, 28

fp\_phase\_correct, 28

fs, 29

ft\_shift, 29

ft\_shift\_mat, 30

gen\_F, 30

gen\_F\_xy, 31

get\_1h\_brain\_basis\_params, 31

get\_1h\_brain\_basis\_params\_v1, 32

get\_1h\_brain\_basis\_params\_v2, 32

get\_acq\_params, 33

get\_dyns, 33

get\_even\_dyns, 34

get\_fh\_dyns, 34

get\_fit\_map, 35

get\_fp, 35

get\_gaussian\_pulse, 36

get\_metab, 36

get\_mol\_names, 37

get\_mol\_params, 37

get\_odd\_dyns, 38

get\_ref, 38

get\_seg\_ind, 39

get\_sh\_dyns, 39

get\_slice, 40

get\_subset, 40

get\_svsvoi, 41

get\_td\_amp, 41

get\_uncoupled\_mol, 42  
 get\_voi\_seg, 42  
 get\_voxel, 43  
  
 hsvd\_filt, 43  
  
 ift\_shift, 44  
 Im.mrs\_data, 44  
 image.mrs\_data, 45  
 int\_spec, 46  
 interleave\_dyns, 46  
 inv\_even\_dyns, 47  
 inv\_odd\_dyns, 47  
 is\_fd, 48  
  
 lb, 48  
 lw2alpha, 49  
 lw2beta, 49  
  
 mat2mrs\_data, 50  
 max\_mrs, 50  
 max\_mrs\_interp, 51  
 mean.mrs\_data, 51  
 mean\_dyn\_blocks, 52  
 mean\_dyn\_pairs, 53  
 mean\_dyns, 52  
 median\_dyns, 53  
 Mod.mrs\_data, 54  
 mrs\_data2basis, 54  
 mrs\_data2mat, 55  
 mrs\_data2vec, 55  
 mvfftshift, 56  
  
 N, 56  
 n2coord, 57  
 Nspec, 57  
  
 peak\_info, 57  
 phase, 58  
 plot.fit\_result, 59  
 plot.mrs\_data, 60  
 plot\_fit\_slice, 61  
 plot\_fit\_slice\_inter, 61  
 plot\_slice\_map, 62  
 plot\_slice\_map\_inter, 62  
 plot\_voi\_overlay, 63  
 plot\_voi\_overlay\_seg, 63  
 ppm, 64  
 print.fit\_result, 64  
 print.mrs\_data, 65  
  
 qn\_states, 65  
  
 rats, 66  
 Re.mrs\_data, 66  
 read\_basis, 67  
 read\_lcm\_coord, 67  
 read\_mrs, 68  
 read\_mrs\_dpt, 69  
 read\_mrs\_tqn, 69  
 read\_tqn\_fit, 70  
 read\_tqn\_result, 71  
 rep\_array\_dim, 71  
 rep\_dyn, 72  
 rep\_mrs, 72  
 resample\_voi, 73  
 rm\_dyns, 73  
  
 scale\_amp\_molal\_pvc, 74  
 scale\_amp\_molar, 74  
 scale\_amp\_ratio, 75  
 scale\_amp\_water\_ratio, 75  
 seconds, 76  
 seq\_cpmg\_ideal, 76  
 seq\_mega\_press\_ideal, 77  
 seq\_press\_ideal, 77  
 seq\_pulse\_acquire, 78  
 seq\_pulse\_acquire\_31p, 79  
 seq\_slaser\_ideal, 79  
 seq\_spin\_echo\_ideal, 80  
 seq\_spin\_echo\_ideal\_31p, 80  
 seq\_steam\_ideal, 81  
 set\_def\_acq\_paras, 81  
 set\_lcm\_cmd, 82  
 set\_ref, 82  
 set\_td\_pts, 82  
 set\_tqn\_cmd, 83  
 shift, 83  
 sim\_basis, 84  
 sim\_basis\_1h\_brain, 84  
 sim\_basis\_1h\_brain\_press, 85  
 sim\_basis\_tqn, 85  
 sim\_brain\_1h, 86  
 sim\_mol, 87  
 sim\_noise, 87  
 sim\_resonances, 88  
 spant (spant-package), 6  
 spant-package, 6  
 spant\_mpress\_drift, 89  
 spin\_sys, 89

spm\_pve2categorical, 90  
stackplot, 90  
stackplot.fit\_result, 91  
stackplot.mrs\_data, 92  
sum\_coils, 93  
sum\_dyns, 93  
  
td2fd, 94  
td\_conv\_filt, 95  
tdsr, 94  
  
varpro\_3\_para\_opts, 95  
varpro\_opts, 96  
vec2mrs\_data, 97  
  
write\_basis, 97  
write\_basis\_tqn, 98  
write\_mrs\_dpt\_v2, 98  
write\_mrs\_lcm\_raw, 99  
  
zero\_nzoc, 99  
zf, 100  
zf\_xy, 100