

# Package ‘spant’

September 14, 2023

**Type** Package

**Title** MR Spectroscopy Analysis Tools

**Version** 2.15.0

**Date** 2023-09-14

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

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

**License** GPL-3

**RoxygenNote** 7.2.3

**NeedsCompilation** yes

**LazyData** yes

**Depends** R (>= 2.10)

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

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

**VignetteBuilder** knitr

**Encoding** UTF-8

**Language** en-GB

**URL** <https://martin3141.github.io/spant/>,  
<https://github.com/martin3141/spant/>

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

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

**Repository** CRAN

**Date/Publication** 2023-09-14 20:20:02 UTC

**R topics documented:**

spant-package . . . . .	8
abfit_opts . . . . .	9
abfit_opts_v1_9_0 . . . . .	13
acquire . . . . .	13
add_noise . . . . .	14
align . . . . .	14
apodise_xy . . . . .	15
append_basis . . . . .	16
append_coils . . . . .	16
append_dyns . . . . .	17
apply_axes . . . . .	17
apply_mrs . . . . .	18
apply_pulse . . . . .	18
Arg.mrs_data . . . . .	19
array2mrs_data . . . . .	19
auto_phase . . . . .	20
back_extrap_ar . . . . .	21
basis2mrs_data . . . . .	21
bbase . . . . .	22
bc_als . . . . .	23
bc_constant . . . . .	23
bc_gauss . . . . .	24
bc_poly . . . . .	24
bc_spline . . . . .	25
beta2lw . . . . .	25
bin_spec . . . . .	26
calc_coil_noise_cor . . . . .	26
calc_coil_noise_sd . . . . .	27
calc_ed_from_lambda . . . . .	27
calc_peak_info_vec . . . . .	28
calc_sd_poly . . . . .	28
calc_spec_diff . . . . .	29
calc_spec_snr . . . . .	29
check_lcm . . . . .	30
check_tqn . . . . .	30
circ_mask . . . . .	31
coherence_filter . . . . .	31
collapse_to_dyns . . . . .	32
comb_coils . . . . .	32
comb_fit_list_fit_tables . . . . .	33
comb_fit_list_result_tables . . . . .	34
comb_fit_tables . . . . .	35
comb_metab_ref . . . . .	35
Conj.mrs_data . . . . .	36
conv_mrs . . . . .	36
crop_basis . . . . .	37

crop_spec . . . . .	37
crop_td_pts . . . . .	38
crop_td_pts_end . . . . .	38
crop_td_pts_pot . . . . .	39
crop_xy . . . . .	39
crossprod_3d . . . . .	40
decimate_mrs_fd . . . . .	40
decimate_mrs_td . . . . .	41
deconv_mrs . . . . .	41
def_acq_paras . . . . .	42
def_fs . . . . .	43
def_ft . . . . .	43
def_N . . . . .	43
def_nuc . . . . .	44
def_ref . . . . .	44
dicom_reader . . . . .	44
diff_mrs . . . . .	45
downsample_mrs_fd . . . . .	46
downsample_mrs_td . . . . .	46
ecc . . . . .	47
elliptical_mask . . . . .	47
est_noise_sd . . . . .	48
fd2td . . . . .	48
fd_conv_filt . . . . .	49
fd_gauss_smo . . . . .	49
find_mrs_files . . . . .	50
fit_amps . . . . .	50
fit_diags . . . . .	51
fit_mrs . . . . .	51
fit_res2csv . . . . .	53
fit_t1_ti_array . . . . .	53
fit_t1_tr_array . . . . .	54
fit_t2_te_array . . . . .	55
fp_phase . . . . .	55
fp_phase_correct . . . . .	56
fp_scale . . . . .	56
fs . . . . .	57
ft_dyns . . . . .	57
ft_shift . . . . .	58
ft_shift_mat . . . . .	58
gausswin_2d . . . . .	59
gen_F . . . . .	59
gen_F_xy . . . . .	60
gen_I . . . . .	60
get_1h_brain_basis_paras . . . . .	61
get_1h_brain_basis_paras_v1 . . . . .	61
get_1h_brain_basis_paras_v2 . . . . .	62
get_1h_brain_basis_paras_v3 . . . . .	62

get_2d_psf . . . . .	63
get_acq_paras . . . . .	63
get_basis_subset . . . . .	64
get_dyns . . . . .	64
get_even_dyns . . . . .	65
get_fh_dyns . . . . .	65
get_fit_map . . . . .	66
get_fp . . . . .	66
get_guassain_pulse . . . . .	67
get_head_dyns . . . . .	67
get_lcm_cmd . . . . .	68
get_metab . . . . .	68
get_mol_names . . . . .	68
get_mol_paras . . . . .	69
get_mrsi2d_seg . . . . .	69
get_mrsi_voi . . . . .	70
get_mrsi_voxel . . . . .	70
get_mrsi_voxel_xy_psf . . . . .	71
get_mrs_affine . . . . .	71
get_odd_dyns . . . . .	72
get_ref . . . . .	72
get_seg_ind . . . . .	73
get_sh_dyns . . . . .	73
get_slice . . . . .	74
get_subset . . . . .	74
get_svs_voi . . . . .	75
get_tail_dyns . . . . .	76
get_td_amp . . . . .	76
get_tqn_cmd . . . . .	77
get_uncoupled_mol . . . . .	77
get_voi_cog . . . . .	78
get_voi_seg . . . . .	78
get_voi_seg_psf . . . . .	79
get_voxel . . . . .	79
gridplot . . . . .	80
gridplot.mrs_data . . . . .	80
grid_shift_xy . . . . .	81
hsvd . . . . .	81
hsvd_filt . . . . .	82
hsvd_vec . . . . .	83
hz . . . . .	83
ift_shift . . . . .	84
ift_shift_mat . . . . .	84
Im.mrs_data . . . . .	85
image.mrs_data . . . . .	85
img2kspace_xy . . . . .	86
Imzap . . . . .	87
interleave_dyns . . . . .	87

int_spec . . . . .	88
inv_even_dyns . . . . .	88
inv_odd_dyns . . . . .	89
is.def . . . . .	89
is_fd . . . . .	90
kspace2img_xy . . . . .	90
l2_reg . . . . .	91
lb . . . . .	92
lofdc . . . . .	92
lw2alpha . . . . .	93
lw2beta . . . . .	94
make_basis_from_raw . . . . .	94
mask_dyns . . . . .	95
mask_fit_res . . . . .	95
mask_xy . . . . .	96
mask_xy_corners . . . . .	96
mask_xy_ellipse . . . . .	97
mask_xy_mat . . . . .	97
mat2mrs_data . . . . .	98
matexp . . . . .	98
max_mrs . . . . .	99
max_mrs_interp . . . . .	99
mean.list . . . . .	100
mean.mrs_data . . . . .	100
mean_dyns . . . . .	101
mean_dyn_blocks . . . . .	101
mean_dyn_pairs . . . . .	102
mean_mrs_list . . . . .	102
mean_vec_blocks . . . . .	103
median_dyns . . . . .	103
Mod.mrs_data . . . . .	104
mod_td . . . . .	104
mrs_data2basis . . . . .	105
mrs_data2mat . . . . .	105
mrs_data2vec . . . . .	106
mvfftshift . . . . .	106
mviftshift . . . . .	107
n2coord . . . . .	107
Ncoils . . . . .	108
Ndyns . . . . .	108
nifti_flip_lr . . . . .	108
Npts . . . . .	109
Nspec . . . . .	109
Ntrans . . . . .	110
Nx . . . . .	110
Ny . . . . .	110
Nz . . . . .	111
one_page_pdf . . . . .	111

ortho3	112
ortho3_inter	113
peak_info	113
pg_extrap_xy	114
phase	115
plot.fit_result	115
plot.mrs_data	117
plot_bc	119
plot_slice_fit	119
plot_slice_fit_inter	120
plot_slice_map	120
plot_slice_map_inter	121
plot_spec_sd	123
plot_voi_overlay	123
plot_voi_overlay_seg	124
ppm	124
precomp	125
print.fit_result	125
print.mrs_data	126
qn_states	126
rats	127
Re.mrs_data	128
read_basis	129
read_ima_coil_dir	129
read_ima_dyn_dir	130
read_lcm_coord	130
read_mrs	131
read_mrs_tqn	132
read_pulse_pta	133
read_siemens_txt_hdr	133
read_tqn_fit	134
read_tqn_result	134
recon_imag	135
recon_imag_vec	135
recon_twix_2d_mrsi	136
rectangular_mask	136
rep_array_dim	137
rep_dyn	137
rep_mrs	138
resample_basis	138
resample_img	139
resample_voi	139
reslice_to_mrs	140
reson_table2mrs_data	140
re_weighting	141
rm_dyns	141
scale_amp_molal	142
scale_amp_molal_pvc	143

scale_amp_molar . . . . .	144
scale_amp_molar2molal_pvc . . . . .	144
scale_amp_ratio . . . . .	145
scale_amp_ratio_value . . . . .	146
scale_amp_water_ratio . . . . .	146
scale_basis_amp . . . . .	147
scale_basis_from_singlet . . . . .	147
scale_mrs_amp . . . . .	148
scale_spec . . . . .	148
sd . . . . .	149
sd.mrs_data . . . . .	150
seconds . . . . .	150
seq_cpmg_ideal . . . . .	151
seq_mega_press_ideal . . . . .	151
seq_press_ideal . . . . .	152
seq_pulse_acquire . . . . .	153
seq_pulse_acquire_31p . . . . .	153
seq_slaser_ideal . . . . .	154
seq_spin_echo_ideal . . . . .	154
seq_spin_echo_ideal_31p . . . . .	155
seq_steam_ideal . . . . .	155
seq_steam_ideal_cof . . . . .	156
seq_steam_ideal_young . . . . .	156
set_def_acq_paras . . . . .	157
set_lcm_cmd . . . . .	158
set_lw . . . . .	158
set_mask_xy_mat . . . . .	159
set_Ntrans . . . . .	159
set_precomp_mode . . . . .	160
set_precomp_verbose . . . . .	160
set_ref . . . . .	160
set_td_pts . . . . .	161
set_tqn_cmd . . . . .	161
set_tr . . . . .	162
shift . . . . .	162
shift_basis . . . . .	163
sim_basis . . . . .	163
sim_basis_1h_brain . . . . .	164
sim_basis_1h_brain_press . . . . .	165
sim_basis_mm_lip_lcm . . . . .	165
sim_basis_tqn . . . . .	166
sim_brain_1h . . . . .	167
sim_mol . . . . .	168
sim_noise . . . . .	169
sim_resonances . . . . .	169
sim_th_excit_profile . . . . .	171
sim_zero . . . . .	171
smooth_dyns . . . . .	172

sort_basis . . . . .	172
spant_abfit_benchmark . . . . .	173
spant_mpress_drift . . . . .	173
spant_simulation_benchmark . . . . .	174
spec_decomp . . . . .	174
spec_op . . . . .	175
spin_sys . . . . .	175
spm_pve2categorical . . . . .	176
ssp . . . . .	176
stackplot . . . . .	177
stackplot.fit_result . . . . .	177
stackplot.mrs_data . . . . .	179
sub_first_dyn . . . . .	181
sub_mean_dyns . . . . .	181
sum_coils . . . . .	182
sum_dyns . . . . .	182
sum_mrs . . . . .	183
sum_mrs_list . . . . .	183
svs_1h_brain_analysis . . . . .	184
svs_1h_brain_batch_analysis . . . . .	185
td2fd . . . . .	186
tdsr . . . . .	186
td_conv_filt . . . . .	187
tr . . . . .	187
varpro_3_para_opts . . . . .	188
varpro_basic_opts . . . . .	188
varpro_opts . . . . .	189
vec2mrs_data . . . . .	190
write_basis . . . . .	191
write_basis_tqn . . . . .	191
write_mrs . . . . .	192
write_mrs_nifti . . . . .	192
zero_fade_spec . . . . .	193
zero_higher_orders . . . . .	193
zero_td_pts_end . . . . .	194
zf . . . . .	194
zf_xy . . . . .	195

**Index****196**

spant-package

*spant: spectroscopy analysis tools.***Description**

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

## Details

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

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

Full list of vignettes:

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

Full list of functions:

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

An online version of the documentation is available from:

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

## Author(s)

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

Other contributors:

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

## See Also

Useful links:

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

---

abfit\_opts

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

---

## Description

Return a list of options for an ABfit analysis.

## Usage

```
abfit_opts(  
  init_damping = 5,  
  maxiters = 1024,  
  max_shift = 0.078,  
  max_damping = 15,  
  max_phase = 360,  
  lambda = NULL,  
  ppm_left = 4,  
  ppm_right = 0.2,  
)
```

```

zp = TRUE,
bl_ed_pppm = 2,
auto_bl_flex = TRUE,
bl_comps_pppm = 15,
export_sp_fit = FALSE,
max_asym = 0.25,
max_basis_shift = 0.0078,
max_basis_damping = 2,
maxiters_pre = 1000,
algo_pre = "NLOPT_LN_NELDERMEAD",
min_bl_ed_pppm = NULL,
max_bl_ed_pppm = 7,
auto_bl_flex_n = 20,
pre_fit_bl_ed_pppm = 1,
remove_lip_mm_prefit = FALSE,
pre_align = TRUE,
max_pre_align_shift = 0.1,
pre_align_ref_freqs = c(2.01, 3.03, 3.22),
noise_region = c(-0.5, -2.5),
optimal_smooth_criterion = "maic",
aic_smoothing_factor = 5,
anal_jac = TRUE,
pre_fit_ppm_left = 4,
pre_fit_ppm_right = 1.8,
phi1_optim = FALSE,
phi1_init = 0,
max_dphi1 = 0.2,
max_basis_shift_broad = 0.0078,
max_basis_damping_broad = 2,
ahat_calc_method = "lh_pnnls",
prefit_phase_search = TRUE,
freq_reg = NULL,
lb_reg = NULL,
output_all_paras = FALSE,
output_all_paras_raw = FALSE,
input_paras_raw = NULL,
optim_lw_only = FALSE,
optim_lw_only_limit = 20,
lb_init = 0.001
)

```

### Arguments

<code>init_damping</code>	initial value of the Gaussian global damping parameter (Hz). Very poorly shimmed or high field data may benefit from a larger value.
<code>maxiters</code>	The maximum number of iterations to run for the detailed fit.
<code>max_shift</code>	The maximum allowable shift to be applied in the optimisation phase of fitting (ppm).

max_damping	maximum permitted value of the global damping parameter (Hz).
max_phase	the maximum absolute permitted value of the global zero-order phase term (degrees). Note, the prefit_phase_search option is not constrained by this term.
lambda	manually set the the baseline smoothness parameter.
ppm_left	downfield frequency limit for the fitting range (ppm).
ppm_right	upfield frequency limit for the fitting range (ppm).
zp	zero pad the data to twice the original length before fitting.
bl_ed_pppm	manually set the the baseline smoothness parameter (ED per ppm).
auto_bl_flex	automatically determine the level of baseline smoothness.
bl_comps_pppm	spline basis density (signals per ppm).
export_sp_fit	add the fitted spline functions to the fit result.
max_asym	maximum allowable value of the asymmetry parameter.
max_basis_shift	maximum allowable frequency shift for individual basis signals (ppm).
max_basis_damping	maximum allowable Lorentzian damping factor for individual basis signals (Hz).
maxiters_pre	maximum iterations for the coarse (pre-)fit.
algo_pre	optimisation method for the coarse (pre-)fit.
min_bl_ed_pppm	minimum value for the candidate baseline flexibility analyses (ED per ppm).
max_bl_ed_pppm	minimum value for the candidate baseline flexibility analyses (ED per ppm).
auto_bl_flex_n	number of candidate baseline analyses to perform.
pre_fit_bl_ed_pppm	level of baseline flexibility to use in the coarse fitting stage of the algorithm (ED per ppm).
remove_lip_mm_prefit	remove broad signals in the coarse fitting stage of the algorithm.
pre_align	perform a pre-alignment step before coarse fitting.
max_pre_align_shift	maximum allowable shift in the pre-alignment step (ppm).
pre_align_ref_freqs	a vector of prominent spectral frequencies used in the pre-alignment step (ppm).
noise_region	spectral region to estimate the noise level (ppm).
optimal_smooth_criterion	method to determine the optimal smoothness.
aic_smoothing_factor	modification factor for the AIC calculation.
anal_jac	use a analytical approximation to the jacobian in the detailed fitting stage.
pre_fit_ppm_left	downfield frequency limit for the fitting range in the coarse fitting stage of the algorithm (ppm).

pre_fit_ppm_right	upfield frequency limit for the fitting range in the coarse fitting stage of the algorithm (ppm).
phi1_optim	apply and optimise a frequency dependant phase term.
phi1_init	initial value for the frequency dependant phase term (ms).
max_dphi1	maximum allowable change from the initial frequency dependant phase term (ms).
max_basis_shift_broad	maximum allowable shift for broad signals in the basis (ppm). Determined based on their name beginning with Lip or MM.
max_basis_damping_broad	maximum allowable Lorentzian damping for broad signals in the basis (Hz). Determined based on their name beginning with Lip or MM.
ahat_calc_method	method to calculate the metabolite amplitudes. May be one of: "lh_pnnls" or "ls".
prefit_phase_search	perform a 1D search for the optimal phase in the prefit stage of the algorithm.
freq_reg	frequency shift parameter.
lb_reg	individual line broadening parameter.
output_all_paras	include more fitting parameters in the fit table, e.g. individual shift and damping factors for each basis set element.
output_all_paras_raw	include raw fitting parameters in the fit table. For advanced diagnostic use only.
input_paras_raw	input raw fitting parameters. For advanced diagnostic use only.
optim_lw_only	optimize the global line-broadening term only.
optim_lw_only_limit	limits for the line-breading term as a percentage of the starting value when optim_lw_only is TRUE.
lb_init	initial Lorentzian line broadening value for the individual basis signals. Setting to 0 will clash with the minimum allowable value (eg hard constraint) during the detailed fit.

**Value**

full list of options.

**Examples**

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

---

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

---

**Description**

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

**Usage**

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

**Arguments**

... arguments passed to [abfit\\_opts](#).

**Value**

full list of options.

---

acquire	<i>Simulate pulse sequence acquisition.</i>
---------	---

---

**Description**

Simulate pulse sequence acquisition.

**Usage**

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

**Arguments**

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

**Value**

a list of resonance amplitudes and frequencies.

---

add_noise	<i>Add noise to an mrs_data object.</i>
-----------	---

---

**Description**

Add noise to an mrs\_data object.

**Usage**

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

**Arguments**

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

**Value**

mrs\_data object with additive normally distributed noise.

---

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_freq = 4.65,  
  ref_amp = 1,  
  zf_factor = 2,  
  lb = 2,  
  max_shift = 20,  
  ret_df = FALSE,  
  mean_dyns = FALSE  
)
```

**Arguments**

mrs_data	data to be aligned.
ref_freq	reference frequency in ppm units. More than one frequency may be specified.
ref_amp	amplitude value for the reference signal. More than one value may be specified to match the number of ref_freq signals.
zf_factor	zero filling factor to increase alignment resolution.
lb	line broadening to apply to the reference signal.
max_shift	maximum allowable shift in Hz.
ret_df	return frequency shifts in addition to aligned data (logical).
mean_dyns	align the mean spectrum and apply the same shift to each dynamic.

**Value**

aligned data object.

---

apodise\_xy

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

apodised data.

---

append_basis	<i>Combine a pair of basis set objects.</i>
--------------	---

---

**Description**

Combine a pair of basis set objects.

**Usage**

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

**Arguments**

basis_a	first basis.
basis_b	second basis.

**Value**

combined basis set object.

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

---

append_dyns	<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\_pulse***Simulate an RF pulse on a single spin.*

---

**Description**

Simulate an RF pulse on a single spin.

**Usage**

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

**Arguments**

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

**Value**

density matrix.

---

Arg.mrs_data	<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.

---

array2mrs_data	<i>Convert a 7 dimensional array in into a mrs_data object. The array dimensions should be ordered as : dummy, X, Y, Z, dynamic, coil, FID.</i>
----------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

mrs\_data object.

---

auto_phase	<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 = c(4, 1.8), smo_ppm_sd = 0.05, ret_phase = FALSE)
```

**Arguments**

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

**Value**

MRS data object and phase values (optional).

---

back_extrap_ar	<i>Back extrapolate time-domain data points using an autoregressive model.</i>
----------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

back extrapolated data.

---

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

**Arguments**

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

**Value**

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

---

bbase	<i>Generate a spline basis, slightly adapted from : "Splines, knots, and penalties", Eilers 2010.</i>
-------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

spline basis as a matrix.

---

bc_als	<i>Baseline correction using the ALS method.</i>
--------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

baseline corrected data.

---

bc_constant	<i>Remove a constant baseline offset based on a reference spectral region.</i>
-------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

baseline corrected data.

---

bc_gauss	<i>Apply and subtract a Gaussian smoother in the spectral domain.</i>
----------	---

---

**Description**

Apply and subtract a Gaussian smoother in the spectral domain.

**Usage**

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

**Arguments**

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

**Value**

smoother subtracted data.

---

bc_poly	<i>Fit and subtract a polynomial to each spectrum in a dataset.</i>
---------	---

---

**Description**

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

**Usage**

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

**Arguments**

mrs_data	mrs_data object.
p_deg	polynomial degree.

**Value**

polynomial subtracted data.

---

bc_spline	<i>Fit and subtract a smoothing spline to each spectrum in a dataset.</i>
-----------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

smoothing spline subtracted data.

---

beta2lw	<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.

---

bin_spec	<i>Bin equally spaced spectral regions.</i>
----------	---

---

**Description**

Bin equally spaced spectral regions.

**Usage**

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

**Arguments**

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

**Value**

binned mrs\_data object.

---

calc_coil_noise_cor	<i>Calculate the noise correlation between coil elements.</i>
---------------------	---

---

**Description**

Calculate the noise correlation between coil elements.

**Usage**

```
calc_coil_noise_cor(noise_data)
```

**Arguments**

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

**Value**

correlation matrix.

---

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

---

**Description**

Calculate the noise standard deviation for each coil element.

**Usage**

```
calc_coil_noise_sd(noise_data)
```

**Arguments**

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

**Value**

array of standard deviations.

---

calc\_ed\_from\_lambda     *Calculate the effective dimensions of a spline smoother from lambda.*

---

**Description**

Calculate the effective dimensions of a spline smoother from lambda.

**Usage**

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

**Arguments**

spline\_basis     spline basis.  
deriv\_mat        derivative matrix.  
lambda            smoothing parameter.

**Value**

the effective dimension value.

---

calc\_peak\_info\_vec      *Calculate the FWHM of a peak from a vector of intensity values.*

---

**Description**

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

**Usage**

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

**Arguments**

data\_pts      input vector.  
interp\_f      interpolation factor to improve the FWHM estimate.

**Value**

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

---

calc\_sd\_poly      *Perform a polynomial fit, subtract and return the standard deviation of the residuals.*

---

**Description**

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

**Usage**

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

**Arguments**

y      array.  
degree      polynomial degree.

**Value**

standard deviation of the fit residuals.

---

calc_spec_diff	<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 the standard deviation of the noise.

**Usage**

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

**Arguments**

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

**Details**

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

**Value**

an array of SNR values.

---

check_lcm	<i>Check LCMModel can be run</i>
-----------	----------------------------------

---

**Description**

Check LCMModel 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()

---

circ_mask	<i>Create a logical circular mask spanning the full extent of an n x n matrix.</i>
-----------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

logical n x n mask matrix.

---

coherence_filter	<i>Zero all coherence orders other than the one supplied as an argument.</i>
------------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

density matrix.

---

collapse_to_dyns	<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(x, rm_masked = FALSE)

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

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

**Arguments**

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

**Value**

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

---

comb_coils	<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,
  scale_method = "sig_noise_sq",
  sum_coils = TRUE,
  noise_region = c(-0.5, -2.5),
```

```

    average_ref_dyns = TRUE,
    ref_pt_index = 1,
    ret_metab_only = FALSE
)

```

### Arguments

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

### Value

MRS data.

---

comb\_fit\_list\_fit\_tables

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

---

### Description

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

### Usage

```

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

```

**Arguments**

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

**Value**

a data frame containing the fit data points.

---

comb\_fit\_list\_result\_tables

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

---

comb_fit_tables	<i>Combine all fitting data points into a single data frame.</i>
-----------------	--

---

**Description**

Combine all fitting data points into a single data frame.

**Usage**

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

**Arguments**

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

**Value**

a data frame containing the fit data points.

---

comb_metab_ref	<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.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_basis	<i>Crop basis_set object based on a frequency range.</i>
------------	--

---

**Description**

Crop basis\_set object based on a frequency range.

**Usage**

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

**Arguments**

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

**Value**

cropped mrs\_data object.

---

crop_spec	<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, 0.2), scale = "ppm")
```

**Arguments**

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

**Value**

cropped mrs\_data object.

---

crop_td_pts	<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.

---

crop_td_pts_end	<i>Crop mrs_data object data points at the end of the FID.</i>
-----------------	--

---

**Description**

Crop mrs\_data object data points at the end of the FID.

**Usage**

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

**Arguments**

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

**Value**

cropped mrs\_data object.

---

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

---

**Description**

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

**Usage**

```
crop_td_pts_pot(mrs_data)
```

**Arguments**

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

**Value**

cropped mrs\_data object.

---

crop_xy	<i>Crop an MRSI dataset in the x-y direction</i>
---------	--

---

**Description**

Crop an MRSI dataset in the x-y direction

**Usage**

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

**Arguments**

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

**Value**

selected subset of MRS data.

---

crossprod_3d	<i>Compute the vector cross product between vectors <math>x</math> and <math>y</math>. Adapted from <a href="http://stackoverflow.com/questions/15162741/what-is-rs-crossproduct-function">http://stackoverflow.com/questions/15162741/what-is-rs-crossproduct-function</a></i>
--------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

vector cross product of  $x$  and  $y$ .

---

decimate_mrs_fd	<i>Decimate an MRS signal to half the original sampling frequency by filtering in the frequency domain before down sampling.</i>
-----------------	--

---

**Description**

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

**Usage**

```
decimate_mrs_fd(mrs_data)
```

**Arguments**

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

**Value**

decimated data at half the original sampling frequency.

---

decimate_mrs_td	<i>Decimate an MRS signal by filtering in the time domain before downsampling.</i>
-----------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

decimated data.

---

deconv_mrs	<i>Deconvolve two MRS data objects.</i>
------------	---

---

**Description**

Deconvolve two MRS data objects.

**Usage**

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

**Arguments**

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

**Value**

deconvolved data.

---

def_acq_paras	<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"),  
    nuc = getOption("spant.def_nuc")  
)
```

### Arguments

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

### Value

A list containing the following elements:

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

---

def\_fs                      *Return the default sampling frequency in Hz.*

---

**Description**

Return the default sampling frequency in Hz.

**Usage**

def\_fs()

**Value**

sampling frequency in Hz.

---

def\_ft                      *Return the default transmitter frequency in Hz.*

---

**Description**

Return the default transmitter frequency in Hz.

**Usage**

def\_ft()

**Value**

transmitter frequency in Hz.

---

def\_N                      *Return the default number of data points in the spectral dimension.*

---

**Description**

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

**Usage**

def\_N()

**Value**

number of data points in the spectral dimension.

def\_nuc *Return the default nucleus.*

---

**Description**

Return the default nucleus.

**Usage**

```
def_nuc()
```

**Value**

number of data points in the spectral dimension.

---

def\_ref *Return the default reference value for ppm scale.*

---

**Description**

Return the default reference value for ppm scale.

**Usage**

```
def_ref()
```

**Value**

reference value for ppm scale.

---

dicom\_reader *A very simple DICOM reader.*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

---

diff_mrs	<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.

---

downsample_mrs_fd	<i>Downsample an MRS signal by a factor of 2 using an FFT "brick-wall" filter.</i>
-------------------	--

---

**Description**

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

**Usage**

```
downsample_mrs_fd(mrs_data)
```

**Arguments**

mrs\_data          MRS data object.

**Value**

downsampled data.

---

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

---

**Description**

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

**Usage**

```
downsample_mrs_td(mrs_data)
```

**Arguments**

mrs\_data          MRS data object.

**Value**

downsampled data.

---

ecc                                    *Eddy current correction.*

---

### Description

Apply eddy current correction using the Klose method.

### Usage

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

### Arguments

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

### Details

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

### Value

corrected data in the time domain.

---

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

---

### Description

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

### Usage

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

### Arguments

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

**Value**

logical mask matrix with dimensions fov\_yN x fov\_xN.

---

est_noise_sd	<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.

---

fd_gauss_smo	<i>Apply a Gaussian smoother in the spectral domain.</i>
--------------	--

---

**Description**

Apply a Gaussian smoother in the spectral domain.

**Usage**

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

**Arguments**

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

**Value**

spectrally smoothed data.

---

find_mrs_files	<i>Find valid MRS data files recursively from a directory path.</i>
----------------	---

---

**Description**

Find valid MRS data files recursively from a directory path.

**Usage**

```
find_mrs_files(dir)
```

**Arguments**

dir                    a directory path.

**Value**

a vector of valid MRS data files.

---

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

---

**Description**

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

**Usage**

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

**Arguments**

x                    fit\_result object.  
inc\_index            include columns for the voxel index.  
sort\_names           sort the basis set names alphabetically.  
append\_common\_1h\_comb    append commonly used 1H metabolite combinations eg tNAA = NAA + NAAG.

**Value**

a dataframe of amplitudes.

---

fit_diags	<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\_cmd and set\_lcm\_cmd (respectively) need to be used to specify the location of these software packages.

**Usage**

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

**Arguments**

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

**Details**

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

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

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

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

**Value**

MRS analysis object.

**Examples**

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

## End(Not run)
```

---

fit_res2csv	<i>Write fit results table to a csv file.</i>
-------------	---

---

**Description**

Write fit results table to a csv file.

**Usage**

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

**Arguments**

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

---

fit_t1_ti_array	<i>Fit a T1 recovery curve, from multiple TIs, to a set of amplitudes.</i>
-----------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

---

fit_t1_tr_array	<i>Fit a T1 recovery curve, from multiple TRs, to a set of amplitudes.</i>
-----------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

---

fit_t2_te_array	<i>Fit a T2 relaxation curve, from multiple TEs, to a set of amplitudes.</i>
-----------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

---

fp_phase	<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.

---

fp_scale	<i>Scale the first time-domain data point in an mrs_data object.</i>
----------	--

---

**Description**

Scale the first time-domain data point in an mrs\_data object.

**Usage**

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

**Arguments**

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

**Value**

scaled mrs\_data object.

---

fs	<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_dyns	<i>Apply the Fourier transform over the dynamic dimension.</i>
---------	--

---

**Description**

Apply the Fourier transform over the dynamic dimension.

**Usage**

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

**Arguments**

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

**Value**

transformed MRS data.

---

ft_shift	<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.

---

gausswin_2d	<i>Create a two dimensional Gaussian window function stored as a matrix.</i>
-------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

matrix with dimensions fov\_yN x fov\_xN.

---

gen_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`*Generate the Fxy product operator with a specified phase.*

---

**Description**

Generate the Fxy product operator with a specified phase.

**Usage**

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

**Arguments**

<code>sys</code>	spin system object.
<code>phase</code>	phase angle in degrees.
<code>detect</code>	detection nuclei.

**Value**

product operator matrix.

---

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

---

**Description**

Generate the I product operator for a single spin.

**Usage**

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

**Arguments**

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

**Value**

I product operator matrix.

---

get\_1h\_brain\_basis\_paras

*Return a list of mol\_parameter objects suitable for 1H brain MRS analyses.*

---

**Description**

Return a list of mol\_parameter objects suitable for 1H brain MRS analyses.

**Usage**

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

**Arguments**

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

**Value**

list of mol\_parameter objects.

---

get\_1h\_brain\_basis\_paras\_v1

*Return a list of mol\_parameter objects suitable for 1H brain MRS analyses.*

---

**Description**

Return a list of mol\_parameter objects suitable for 1H brain MRS analyses.

**Usage**

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

**Arguments**

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

**Value**

list of mol\_parameter objects.

---

get\_1h\_brain\_basis\_paras\_v2

*Return a list of mol\_parameter objects suitable for 1H brain MRS analyses.*

---

**Description**

Return a list of mol\_parameter objects suitable for 1H brain MRS analyses.

**Usage**

get\_1h\_brain\_basis\_paras\_v2(ft, metab\_lw = NULL, lcm\_compat = FALSE)

**Arguments**

ft transmitter frequency in Hz.  
metab\_lw linewidth of metabolite signals (Hz).  
lcm\_compat when TRUE, lipid, MM and -CrCH molecules will be excluded from the output.

**Value**

list of mol\_parameter objects.

---

get\_1h\_brain\_basis\_paras\_v3

*Return a list of mol\_parameter objects suitable for 1H brain MRS analyses.*

---

**Description**

Return a list of mol\_parameter objects suitable for 1H brain MRS analyses.

**Usage**

get\_1h\_brain\_basis\_paras\_v3(ft, metab\_lw = NULL, lcm\_compat = FALSE)

**Arguments**

ft transmitter frequency in Hz.  
metab\_lw linewidth of metabolite signals (Hz).  
lcm\_compat when TRUE, lipid, MM and -CrCH molecules will be excluded from the output.

**Value**

list of mol\_parameter objects.

---

get_2d_psf	<i>Get the point spread function (PSF) for a 2D phase encoded MRSI scan.</i>
------------	--

---

**Description**

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

**Usage**

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

**Arguments**

FOV	field of view in mm.
mat_size	acquisition matrix size (not interpolated).
sampling	can be either "circ" for circular or "rect" for rectangular.
hamming	should Hamming k-space weighting be applied (default FALSE).
ensure_odd	add 1mm to the FOV when required to ensure the output pdf has odd dimensions. Required when using get_mrsi2d_seg.

**Value**

A matrix of the PSF with 1mm resolution.

---

get_acq_paras	<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\_basis\_subset      *Return a subset of the input basis.*

---

**Description**

Return a subset of the input basis.

**Usage**

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

**Arguments**

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

**Value**

a subset of the input basis.

---

get\_dyns      *Extract a subset of dynamic scans.*

---

**Description**

Extract a subset of dynamic scans.

**Usage**

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

**Arguments**

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

**Value**

MRS data containing the subset of requested dynamics.

---

get_even_dyns	<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\_head\_dyns      *Return the first scans of a dynamic series.*

---

**Description**

Return the first scans of a dynamic series.

**Usage**

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

**Arguments**

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

**Value**

first scans of a dynamic series.

---

get_lcm_cmd	<i>Print the command to run the LCModel command-line program.</i>
-------------	---

---

**Description**

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

**Usage**

```
get_lcm_cmd()
```

---

get_metab	<i>Extract the metabolite component from an mrs_data object.</i>
-----------	--

---

**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_mrsi2d_seg	<i>Calculate the partial volume estimates for each voxel in a 2D MRSI dataset.</i>
----------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

a data frame of partial volume estimates and individual segmentation maps.

---

get\_mrsi\_voi                      *Generate a MRSI VOI from an mrs\_data object.*

---

### Description

Generate a MRSI VOI from an mrs\_data object.

### Usage

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

### Arguments

mrs_data	MRS data.
target_mri	optional image data to match the intended volume space.
map	optional voi intensity map.
ker	kernel to rescale the map data to the target_mri. Default value is mmand::boxKernel(), use mmand::mnKernel() for a smoothed map.

### Value

volume data as a nifti object.

---

get\_mrsi\_voxel                      *Generate a MRSI voxel from an mrs\_data object.*

---

### Description

Generate a MRSI voxel from an mrs\_data object.

### Usage

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

### Arguments

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

### Value

volume data as a nifti object.

---

get\_mrsi\_voxel\_xy\_psf *Generate a MRSI voxel PSF from an mrs\_data object.*

---

**Description**

Generate a MRSI voxel PSF from an mrs\_data object.

**Usage**

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

**Arguments**

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

**Value**

volume data as a nifti object.

---

get\_mrs\_affine *Generate an affine for nifti generation.*

---

**Description**

Generate an affine for nifti generation.

**Usage**

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

**Arguments**

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

**Value**

affine matrix.

---

get_odd_dyns	<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,  
  fd_set = NULL,  
  td_set = NULL  
)
```

**Arguments**

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

**Value**

selected subset of MRS data.

---

get\_svs\_voi

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

volume data as a nifti object.

---

get_tail_dyns	<i>Return the last scans of a dynamic series.</i>
---------------	---

---

**Description**

Return the last scans of a dynamic series.

**Usage**

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

**Arguments**

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

**Value**

last scans of a dynamic series.

---

get_td_amp	<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, method = "poly")
```

**Arguments**

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

**Value**

array of amplitudes.

---

get_tqn_cmd	<i>Print the command to run the TARQUIN command-line program.</i>
-------------	---

---

**Description**

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

**Usage**

```
get_tqn_cmd()
```

---

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,
    full_name = NULL
)
```

**Arguments**

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

**Value**

mol\_parameters object.

---

get_voi_cog	<i>Calculate the centre of gravity for an image containing 0 and 1's.</i>
-------------	---

---

**Description**

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

**Usage**

```
get_voi_cog(voi)
```

**Arguments**

voi            nifti object.

**Value**

triplet of x,y,z coordinates.

---

get_voi_seg	<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_voi_seg_psf	<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_psf(psf, mri_seg)
```

**Arguments**

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

**Value**

a vector of partial volumes expressed as percentages.

---

get_voxel	<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.

---

gridplot	<i>Arrange spectral plots in a grid.</i>
----------	--

---

**Description**

Arrange spectral plots in a grid.

**Usage**

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

**Arguments**

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

---

gridplot.mrs_data	<i>Arrange spectral plots in a grid.</i>
-------------------	--

---

**Description**

Arrange spectral plots in a grid.

**Usage**

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

**Arguments**

x	object of class <code>mrs_data</code> .
rows	number of grid rows.
cols	number of grid columns.
mar	option to adjust the plot margins. See <code>?par</code> .
oma	outer margin area.

bty                   option to draw a box around the plot. See ?par.  
 restore\_def\_par       restore default plotting par values after the plot has been made.  
 ...                   other arguments to pass to the plot method.

---

grid\_shift\_xy                   *Grid shift MRSI data in the x/y dimension.*

---

### Description

Grid shift MRSI data in the x/y dimension.

### Usage

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

### Arguments

mrs\_data              MRSI data in the spatial domain.  
 x\_shift               shift to apply in the x-direction in units of voxels.  
 y\_shift               shift to apply in the y-direction in units of voxels.

### Value

shifted data.

---

hsvd                       *HSVD of an mrs\_data object.*

---

### Description

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

### Usage

```
hsvd(mrs_data, comps = 40, irlba = TRUE, max_damp = 10)
```

### Arguments

mrs\_data              mrs\_data object to be decomposed.  
 comps                 number of Lorentzian components to use for modelling.  
 irlba                 option to use irlba SVD (logical).  
 max\_damp             maximum allowable damping factor.

**Value**

basis matrix and signal table.

---

hsvd_filt	<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 = 40,
  irlba = TRUE,
  max_damp = 10,
  scale = "hz",
  return_model = FALSE
)
```

**Arguments**

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

**Value**

filtered data or model depending on the return\_model argument.

---

hsvd_vec	<i>HSVD of a complex vector.</i>
----------	----------------------------------

---

**Description**

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

**Usage**

```
hsvd_vec(y, fs, comps = 40, irlba = TRUE, max_damp = 0)
```

**Arguments**

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

**Value**

basis matrix and signal table.

---

hz	<i>Return the frequency scale of an MRS dataset in Hz.</i>
----	--

---

**Description**

Return the frequency scale of an MRS dataset in Hz.

**Usage**

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

**Arguments**

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

**Value**

frequency scale.

---

ift_shift	<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.

---

ift_shift_mat	<i>Perform an ifft and ifftshift on a matrix with each column replaced by its shifted ifft.</i>
---------------	---

---

**Description**

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

**Usage**

```
ift_shift_mat(mat_in)
```

**Arguments**

mat\_in            matrix input.

**Value**

output matrix.

---

Im.mrs_data	<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,
  plot_dim = NULL,
  x_pos = NULL,
  y_pos = NULL,
  z_pos = NULL,
  dyn = 1,
  coil = 1,
  restore_def_par = TRUE,
  y_ticks = NULL,
  vline = NULL,
  hline = NULL,
  ...
)
```

**Arguments**

x	object of class <code>mrs_data</code> .
xlim	the range of values to display on the x-axis, eg <code>xlim = c(4,1)</code> .
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 <code>viridis</code> .
plot_dim	the dimension to display on the y-axis, can be one of: "dyn", "x", "y", "z", "coil" or NULL. If NULL (the default) all spectra will be collapsed into the dynamic dimension and displayed.
x_pos	the x index to plot.
y_pos	the y index to plot.
z_pos	the z index to plot.
dyn	the dynamic index to plot.
coil	the coil element number to plot.
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 <code>vline</code> .
hline	draw a horizontal line at the value of <code>hline</code> .
...	other arguments to pass to the plot method.

---



---

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


---

**Description**

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

**Usage**

```
img2kspace_xy(mrs_data)
```

**Arguments**

mrs_data	2D MRSI data.
----------	---------------

**Value**

k-space data.

---

Imzap	<i>Complex rounding function taken from complexplus package to reduce the number of spant dependencies.</i>
-------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

---

interleave_dyns	<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**

See spec\_op function for a more complete set of spectral operations.

**Usage**

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

**Arguments**

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

**Value**

an array of integral values.

---

inv_even_dyns	<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.def	<i>Check if an object is defined, which is the same as being not NULL.</i>
--------	--

---

**Description**

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

**Usage**

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

**Arguments**

x	object to test for being NULL.
---	--------------------------------

**Value**

logical value.

---

is_fd	<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.

---

kspace2img_xy	<i>Transform 2D MRSI data from k-space to image space in the x-y direction.</i>
---------------	---

---

**Description**

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

**Usage**

```
kspace2img_xy(mrs_data)
```

**Arguments**

mrs_data	2D MRSI data.
----------	---------------

**Value**

MRSI data in image space.

---

`l2_reg`*Perform l2 regularisation artefact suppression.*

---

**Description**

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

**Usage**

```
l2_reg(  
  mrs_data,  
  thresh = 0.05,  
  b = 1e-11,  
  A = NA,  
  xlim = NA,  
  thresh_xlim = NULL,  
  A_append = NULL,  
  ret_norms = FALSE  
)
```

**Arguments**

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

**Value**

l2 reconstructed `mrs_data` object.

---

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 'list'
lb(x, lb, lg = 1)

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

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

**Arguments**

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

**Value**

line-broadened data.

---

lofdc	<i>Correct linear frequency drift.</i>
-------	--

---

**Description**

Correct linear frequency drift.

**Usage**

```
lofdc(
  mrs_data,
  max_hz_s = 0.1,
  tr = NULL,
  ret_corr_only = TRUE,
  outlier_thresh = 3,
```

```

    xlim = c(4, 0.5),
    order = 1
)

```

### Arguments

**mrs\_data** MRS data to be corrected.  
**max\_hz\_s** the maximum drift rate to search over.  
**tr** mrs\_data repetition time.  
**ret\_corr\_only** return the corrected mrs\_data object only.  
**outlier\_thresh** threshold to remove outliers.  
**xlim** spectral width (in ppm) to evaluate outliers.  
**order** correction order.

### Value

drift corrected mrs\_data object.

---

lw2alpha	<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 * \text{beta})</math>.</i>
---------	---

---

**Description**

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

**Usage**

```
lw2beta(lw)
```

**Arguments**

lw	linewidth in Hz.
----	------------------

**Value**

beta damping value.

---

make_basis_from_raw	<i>Make a basis-set object from a directory containing LCModel formatted RAW files.</i>
---------------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

a basis-set object.

---

mask_dyns	<i>Mask an MRS dataset in the dynamic dimension.</i>
-----------	--

---

**Description**

Mask an MRS dataset in the dynamic dimension.

**Usage**

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

**Arguments**

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

**Value**

masked dataset.

---

mask_fit_res	<i>Mask fit result spectra depending on a vector of bool values.</i>
--------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

a masked fit result object.

---

mask_xy	<i>Mask an MRSI dataset in the x-y direction</i>
---------	--

---

**Description**

Mask an MRSI dataset in the x-y direction

**Usage**

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

**Arguments**

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

**Value**

masked MRS data.

---

mask_xy_corners	<i>Mask the four corners of an MRSI dataset in the x-y plane.</i>
-----------------	---

---

**Description**

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

**Usage**

```
mask_xy_corners(mrs_data)
```

**Arguments**

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

**Value**

masked MRS data.

---

mask_xy_ellipse	<i>Mask the voxels outside an elliptical region spanning the MRSI dataset in the x-y plane.</i>
-----------------	---

---

**Description**

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

**Usage**

```
mask_xy_ellipse(mrs_data)
```

**Arguments**

mrs\_data          MRS data object.

**Value**

masked MRS data.

---

mask_xy_mat	<i>Mask a 2D MRSI dataset in the x-y dimension.</i>
-------------	---

---

**Description**

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

**Usage**

```
mask_xy_mat(mrs_data, mask, value = NA)
```

**Arguments**

mrs\_data          MRS data object.

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

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

**Value**

masked dataset.

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

mrs\_data object.

---

matexp	<i>Matrix exponential function taken from complexplus package to reduce the number of spant dependencies.</i>
--------	---

---

**Description**

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

**Usage**

```
matexp(x)
```

**Arguments**

x                    a square complex matrix.

**Value**

the matrix exponential of x.

---

max_mrs	<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.list	<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 'list'  
mean(x, ...)
```

**Arguments**

x                    object of class mrs\_data.  
...                  other arguments to pass to the colMeans function.

**Value**

mean mrs\_data object.

---

mean.mrs_data	<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      *Calculate the pairwise means across a dynamic data set.*

---

**Description**

Calculate the pairwise means across a dynamic data set.

**Usage**

```
mean_dyn_pairs(mrs_data)
```

**Arguments**

mrs\_data      dynamic MRS data.

**Value**

mean dynamic data of adjacent dynamic pairs.

---

mean\_mrs\_list      *Return the mean of a list of mrs\_data objects.*

---

**Description**

Return the mean of a list of mrs\_data objects.

**Usage**

```
mean_mrs_list(mrs_list)
```

**Arguments**

mrs\_list      list of mrs\_data objects.

**Value**

mean mrs\_data object.

---

mean_vec_blocks	<i>Calculate the mean of adjacent blocks in a vector.</i>
-----------------	---

---

**Description**

Calculate the mean of adjacent blocks in a vector.

**Usage**

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

**Arguments**

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

**Value**

vector data averaged in blocks.

---

median_dyns	<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

*Apply Mod operator to an MRS dataset.*

---

**Description**

Apply Mod operator to an MRS dataset.

**Usage**

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

**Arguments**

z                    MRS data.

**Value**

MRS data following Mod operator.

---

mod\_td

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

---

**Description**

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

**Usage**

```
mod_td(mrs_data)
```

**Arguments**

mrs\_data            MRS data input.

**Value**

time-domain modulus of input.

---

mrs_data2basis	<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 column dimension and dynamics in the row dimension.</i>
--------------	---

---

**Description**

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

**Usage**

```
mrs_data2mat(mrs_data, collapse = TRUE)
```

**Arguments**

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

**Value**

MRS data matrix.

---

mrs_data2vec	<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.

---

mvfftshift	<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.

---

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

---

**Description**

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

**Usage**

```
mvfftshift(x)
```

**Arguments**

x                    matrix input.

**Value**

output matrix.

---

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.

---

Ncoils	<i>Return the total number of coil elements in an MRS dataset.</i>
--------	--

---

**Description**

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

**Usage**

```
Ncoils(mrs_data)
```

**Arguments**

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

---

Ndysns	<i>Return the total number of dynamic scans in an MRS dataset.</i>
--------	--

---

**Description**

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

**Usage**

```
Ndysns(mrs_data)
```

**Arguments**

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

---

nifti_flip_lr	<i>Flip the x data dimension order of a nifti image. This corresponds to flipping MRI data in the left-right direction, assuming the data is saved in neurological format (can check with fsorient program).</i>
---------------	--

---

**Description**

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

**Usage**

```
nifti_flip_lr(x)
```

**Arguments**

x                    nifti object to be processed.

**Value**

nifti object with reversed x data direction.

---

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

---

**Description**

Return the number of data points in an MRS dataset.

**Usage**

Npts(mrs\_data)

**Arguments**

mrs\_data            MRS data.

**Value**

number of data points.

---

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

---

**Description**

Return the total number of spectra in an MRS dataset.

**Usage**

Nspec(mrs\_data)

**Arguments**

mrs\_data            MRS data.

---

Ntrans	<i>Return the total number of acquired transients for an MRS dataset.</i>
--------	---

---

**Description**

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

**Usage**

Ntrans(mrs\_data)

**Arguments**

mrs\_data      MRS data.

---

Nx	<i>Return the total number of x locations in an MRS dataset.</i>
----	--

---

**Description**

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

**Usage**

Nx(mrs\_data)

**Arguments**

mrs\_data      MRS data.

---

Ny	<i>Return the total number of y locations in an MRS dataset.</i>
----	--

---

**Description**

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

**Usage**

Ny(mrs\_data)

**Arguments**

mrs\_data      MRS data.

---

Nz	<i>Return the total number of z locations in an MRS dataset.</i>
----	--

---

**Description**

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

**Usage**

```
Nz(mrs_data)
```

**Arguments**

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

---

one_page_pdf	<i>Export a one-page pdf of a single fit result</i>
--------------	---

---

**Description**

Export a one-page pdf of a single fit result

**Usage**

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

**Arguments**

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

---

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

---

**Description**

Display an orthographic projection plot of a nifti object.

**Usage**

```
ortho3(  
  underlay,  
  overlay = NULL,  
  xyz = NULL,  
  zlim = NULL,  
  zlim_ol = NULL,  
  alpha = 0.7,  
  col_ol = viridisLite::viridis(64),  
  orient_lab = TRUE,  
  rescale = 1,  
  crosshairs = TRUE,  
  ch_lwd = 1,  
  colourbar = TRUE,  
  bg = "black",  
  mar = c(0, 0, 0, 0),  
  smallplot = c(0.63, 0.65, 0.07, 0.42)  
)
```

**Arguments**

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

---

ortho3_inter	<i>Display an interactive orthographic projection plot of a nifti object.</i>
--------------	---

---

**Description**

Display an interactive orthographic projection plot of a nifti object.

**Usage**

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

**Arguments**

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

---

peak_info	<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**

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

**Value**

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

---

pg_extrap_xy	<i>Papoulis-Gerchberg (PG) algorithm method for k-space extrapolation.</i>
--------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

extrapolated mrs\_data object.

---

phase	<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**

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

**Value**

MRS data with applied phase parameters.

---

plot.fit_result	<i>Plot the fitting results of an object of class fit_result.</i>
-----------------	---

---

**Description**

Plot the fitting results of an object of class fit\_result.

**Usage**

```
## S3 method for class 'fit_result'
plot(
  x,
  dyn = 1,
  x_pos = 1,
  y_pos = 1,
  z_pos = 1,
  coil = 1,
  xlim = NULL,
  data_only = FALSE,
  label = NULL,
  plot_sigs = NULL,
```

```

n = NULL,
sub_bl = FALSE,
mar = NULL,
restore_def_par = TRUE,
ylim = NULL,
y_scale = FALSE,
show_grid = TRUE,
grid_nx = NULL,
grid_ny = NA,
...
)

```

### Arguments

x	fit_result object.
dyn	the dynamic index to plot.
x_pos	the x index to plot.
y_pos	the y index to plot.
z_pos	the z index to plot.
coil	the coil element number to plot.
xlim	the range of values to display on the x-axis, eg xlim = c(4,1).
data_only	display only the processed data (logical).
label	character string to add to the top left of the plot window.
plot_sigs	a character vector of signal names to add to the plot.
n	single index element to plot (overrides other indices when given).
sub_bl	subtract the baseline from the data and fit (logical).
mar	option to adjust the plot margins. See ?par.
restore_def_par	restore default plotting par values after the plot has been made.
ylim	range of values to display on the y-axis, eg ylim = c(0,10).
y_scale	option to display the y-axis values (logical).
show_grid	plot gridlines behind the data (logical). Defaults to TRUE.
grid_nx	number of cells of the grid in x and y direction. When NULL the grid aligns with the tick marks on the corresponding default axis (i.e., tickmarks as computed by axTicks). When NA, no grid lines are drawn in the corresponding direction.
grid_ny	as above.
...	further arguments to plot method.

---

`plot.mrs_data`*Plotting method for objects of class mrs\_data.*

---

**Description**

Plotting method for objects of class `mrs_data`.

**Usage**

```
## S3 method for class 'mrs_data'
plot(
  x,
  dyn = 1,
  x_pos = 1,
  y_pos = 1,
  z_pos = 1,
  coil = 1,
  fd = TRUE,
  x_units = NULL,
  xlim = NULL,
  y_scale = FALSE,
  x_ax = TRUE,
  mode = "re",
  lwd = NULL,
  bty = NULL,
  label = "",
  restore_def_par = TRUE,
  mar = NULL,
  xaxis_lab = NULL,
  yaxis_lab = NULL,
  xat = NULL,
  xlabs = TRUE,
  yat = NULL,
  ylabs = TRUE,
  show_grid = TRUE,
  grid_nx = NULL,
  grid_ny = NA,
  col = NULL,
  alpha = NULL,
  bl_lty = NULL,
  hline = NULL,
  hline_lty = 2,
  hline_col = "red",
  ...
)
```

**Arguments**

x	object of class <code>mrs_data</code> .
dyn	the dynamic index to plot.
x_pos	the x index to plot.
y_pos	the y index to plot.
z_pos	the z index to plot.
coil	the coil element number to plot.
fd	display data in the frequency-domain (default), or time-domain (logical).
x_units	the units to use for the x-axis, can be one of: "ppm", "hz", "points" or "seconds".
xlim	the range of values to display on the x-axis, eg <code>xlim = c(4,1)</code> .
y_scale	option to display the y-axis values (logical).
x_ax	option to display the x-axis values (logical).
mode	representation of the complex numbers to be plotted, can be one of: "re", "im", "mod" or "arg".
lwd	plot linewidth.
bty	option to draw a box around the plot. See <code>?par</code> .
label	character string to add to the top left of the plot window.
restore_def_par	restore default plotting par values after the plot has been made.
mar	option to adjust the plot margins. See <code>?par</code> .
xaxis_lab	x-axis label.
yaxis_lab	y-axis label.
xat	x-axis tick label values.
xlabs	x-axis tick labels.
yat	y-axis tick label values.
ylabs	y-axis tick labels.
show_grid	plot gridlines behind the data (logical). Defaults to TRUE.
grid_nx	number of cells of the grid in x and y direction. When NULL the grid aligns with the tick marks on the corresponding default axis (i.e., tickmarks as computed by <code>axTicks</code> ). When NA, no grid lines are drawn in the corresponding direction.
grid_ny	as above.
col	set the line colour, eg <code>col = rgb(0.5, 0.5, 0.5)</code> .
alpha	set the line transparency, eg <code>alpha = 0.5</code> is 50% transparency. Overrides any transparency levels set by <code>col</code> .
bl_lty	linetype for the <code>y = 0</code> baseline trace. A default value NULL results in no baseline being plotted.
hline	add a horizontal line at the specified value.
hline_lty	linetype for the horizontal line.
hline_col	colour for the horizontal line.
...	other arguments to pass to the plot method.

---

plot_bc	<i>Convenience function to plot a baseline estimate with the original data.</i>
---------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

---

plot_slice_fit	<i>Plot a 2D slice from an MRSI fit result object.</i>
----------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

---

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

---

### Description

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

### Usage

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

### Arguments

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

---

plot\_slice\_map            *Plot a slice from a 7 dimensional array.*

---

### Description

Plot a slice from a 7 dimensional array.

**Usage**

```

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

```

**Arguments**

data	7d array of values to be plotted.
zlim	smallest and largest values to be plotted.
mask_map	matching map with logical values to indicate if the corresponding values should be plotted.
mask_cutoff	minimum values to plot (as a percentage of the maximum).
interp	map interpolation factor.
slice	the slice index to plot.
dyn	the dynamic index to plot.
coil	the coil element number to plot.
ref	reference index to plot.
denom	map to use as a denominator.
horizontal	display the colourbar horizontally (logical).

---

plot\_slice\_map\_inter *Plot an interactive slice map from a data array where voxels can be selected to display a corresponding spectrum.*

---

**Description**

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

**Usage**

```

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

```

**Arguments**

<code>mrs_data</code>	spectral data.
<code>map</code>	array of values to be plotted, defaults to the integration of the modulus of the full spectral width.
<code>xlim</code>	spectral region to plot.
<code>slice</code>	the slice index to plot.
<code>zlim</code>	smallest and largest values to be plotted.
<code>mask_map</code>	matching map with logical values to indicate if the corresponding values should be plotted.
<code>denom</code>	map to use as a denominator.
<code>mask_cutoff</code>	minimum values to plot (as a percentage of the maximum).
<code>interp</code>	map interpolation factor.
<code>mode</code>	representation of the complex spectrum to be plotted, can be one of: "re", "im", "mod" or "arg".
<code>y_scale</code>	option to display the y-axis values (logical).
<code>ylim</code>	intensity range to plot.
<code>coil</code>	coil element to plot.
<code>fd</code>	display data in the frequency-domain (default), or time-domain (logical).

---

plot\_spec\_sd                    *Plot the spectral standard deviation.*

---

**Description**

Plot the spectral standard deviation.

**Usage**

```
plot_spec_sd(mrs_data, xlim = NULL, scale_sd = 5, ...)
```

**Arguments**

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

---

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

---

**Description**

Plot a volume as an image overlay.

**Usage**

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

**Arguments**

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

---

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

---

### Description

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

### Usage

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

### Arguments

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

---

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

---

### Description

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

### Usage

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

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

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

### Arguments

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

**Value**

ppm scale.

---

precomp	<i>Save function results to file and load on subsequent calls to avoid repeat computation.</i>
---------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

---

print.fit_result	<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`      *Print a summary of mrs\_data parameters.*

---

**Description**

Print a summary of mrs\_data parameters.

**Usage**

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

**Arguments**

<code>x</code>	mrs_data object.
<code>full</code>	print all parameters (default FALSE).
<code>...</code>	further arguments.

---

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

---

**Description**

Get the quantum coherence matrix for a spin system.

**Usage**

```
qn_states(sys)
```

**Arguments**

<code>sys</code>	spin system object.
------------------	---------------------

**Value**

quantum coherence number matrix.

---

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

---

## Description

Robust Alignment to a Target Spectrum (RATS).

## Usage

```
rats(  
  mrs_data,  
  ref = NULL,  
  xlim = c(4, 0.5),  
  max_shift = 20,  
  p_deg = 2,  
  sp_N = 2,  
  sp_deg = 3,  
  max_t = 0.2,  
  basis_type = "poly",  
  rescale_output = TRUE,  
  phase_corr = TRUE,  
  ret_corr_only = TRUE,  
  zero_freq_shift_t0 = FALSE,  
  remove_freq_outliers = FALSE,  
  freq_outlier_thresh = 3,  
  remove_phase_outliers = FALSE,  
  phase_outlier_thresh = 3,  
  remove_amp_outliers = FALSE,  
  amp_outlier_thresh = 3  
)
```

## Arguments

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

**basis\_type**      may be one of "poly" or "spline".  
**rescale\_output**    rescale the bl\_matched\_spec and bl output to improve consistency between dynamic scans.  
**phase\_corr**      apply phase correction (in addition to frequency). TRUE by default.  
**ret\_corr\_only**    return the corrected mrs\_data object only.  
**zero\_freq\_shift\_t0**  
                     perform a linear fit to the frequency shifts and set the (linearly modeled) shift to be 0 Hz for the first dynamic scan.  
**remove\_freq\_outliers**  
                     remove dynamics based on their frequency shift.  
**freq\_outlier\_thresh**  
                     threshold to remove frequency outliers.  
**remove\_phase\_outliers**  
                     remove dynamics based on their phase shift.  
**phase\_outlier\_thresh**  
                     threshold to remove phase outliers.  
**remove\_amp\_outliers**  
                     remove dynamics based on their amplitude change.  
**amp\_outlier\_thresh**  
                     threshold to remove amplitude outliers.

**Value**

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

---

Re.mrs\_data

*Apply Re operator to an MRS dataset.*

---

**Description**

Apply Re operator to an MRS dataset.

**Usage**

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

**Arguments**

**z**                    MRS data.

**Value**

MRS data following Re operator.

---

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

---

**Description**

Read a basis file in LCModel .basis format.

**Usage**

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

**Arguments**

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

**Value**

basis object.

---

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

---

**Description**

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

**Usage**

```
read_ima_coil_dir(dir, extra = NULL, verbose = FALSE)
```

**Arguments**

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

**Value**

mrs\_data object.

---

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

---

**Description**

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

**Usage**

```
read_ima_dyn_dir(dir, extra = NULL, verbose = FALSE)
```

**Arguments**

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

**Value**

mrs\_data object.

---

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

---

**Description**

Read an LCModel formatted coord file containing fit information.

**Usage**

```
read_lcm_coord(coord_f)
```

**Arguments**

coord_f	path to the coord file.
---------	-------------------------

**Value**

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

---

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

---

**Description**

Read MRS data from a file.

**Usage**

```
read_mrs(
  fname,
  format = NULL,
  ft = NULL,
  fs = NULL,
  ref = NULL,
  n_ref_scans = NULL,
  full_fid = FALSE,
  omit_svs_ref_scans = TRUE,
  verbose = FALSE,
  extra = NULL
)
```

**Arguments**

fname	filename of the dpt format MRS data.
format	string describing the data format. Must be one of the following : "spar_sdat", "rda", "dicom", "twix", "pfile", "list_data", "paravis", "dpt", "lcm_raw", "rds", "nifti", "varian", "jmrui_txt". If not specified, the format will be guessed from the filename extension.
ft	transmitter frequency in Hz (required for list_data format).
fs	sampling frequency in Hz (required for list_data format).
ref	reference value for ppm scale (required for list_data format).
n_ref_scans	override the number of water reference scans detected in the file header (GE p-file only).
full_fid	export all data points, including those before the start of the FID (default = FALSE), TWIX format only.
omit_svs_ref_scans	remove any reference scans sometimes saved in SVS twix data (default = TRUE).
verbose	print data file information (default = FALSE).
extra	an optional data frame to provide additional variables for use in subsequent analysis steps, eg id or grouping variables.

**Value**

MRS data object.

### Examples

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

---

read\_mrs\_tqn                      *Read MRS data using the TARQUIN software package.*

---

### Description

Read MRS data using the TARQUIN software package.

### Usage

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

### Arguments

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

### Value

MRS data object.

### Examples

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

## End(Not run)
```

---

read_pulse_pta	<i>Read a .pta formatted text file compatible with Siemens PulseTool.</i>
----------------	---

---

**Description**

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

**Usage**

```
read_pulse_pta(fname)
```

**Arguments**

fname	pta formatted pulse file name.
-------	--------------------------------

**Value**

pulse waveform and header.

---

read_siemens_txt_hdr	<i>Read the text format header found in Siemens IMA and TWIX data files.</i>
----------------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

a list of parameter values

---

read_tqn_fit	<i>Reader for csv fit results generated by TARQUIN.</i>
--------------	---

---

**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)
```

---

recon_imag	<i>Reconstruct complex time-domain data from the real part of frequency-domain data.</i>
------------	--

---

**Description**

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

**Usage**

```
recon_imag(mrs_data)
```

**Arguments**

mrs\_data          MRS data.

**Value**

reconstructed MRS data.

---

recon_imag_vec	<i>Reconstruct complex time-domain data from the real part of frequency-domain data.</i>
----------------	--

---

**Description**

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

**Usage**

```
recon_imag_vec(data)
```

**Arguments**

data                data points in the frequency domain.

**Value**

reconstructed signal.

---

recon_twix_2d_mrsi	<i>Reconstruct 2D MRSI data from a twix file loaded with read_mrs.</i>
--------------------	--

---

**Description**

Reconstruct 2D MRSI data from a twix file loaded with read\_mrs.

**Usage**

```
recon_twix_2d_mrsi(twix_mrs)
```

**Arguments**

twix_mrs	raw dynamic data.
----------	-------------------

**Value**

reconstructed data.

---

rectangular_mask	<i>Create a rectangular mask stored as a matrix of logical values.</i>
------------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

logical mask matrix with dimensions fov\_yN x fov\_xN.

---

rep_array_dim	<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,
  warn = TRUE
)
```

**Arguments**

mrs_data	MRS data to be replicated.
x_rep	number of x replications.
y_rep	number of y replications.
z_rep	number of z replications.
dyn_rep	number of dynamic replications.
coil_rep	number of coil replications.
warn	print a warning when the data dimensions do not change.

**Value**

replicated data object.

---

resample_basis	<i>Resample a basis-set to match a mrs_data acquisition.</i>
----------------	--

---

**Description**

Resample a basis-set to match a mrs\_data acquisition.

**Usage**

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

**Arguments**

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

**Value**

resampled basis set object.

---

resample_img	<i>Resample an image to match a target image space.</i>
--------------	---

---

**Description**

Resample an image to match a target image space.

**Usage**

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

**Arguments**

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

**Value**

resampled image data as a nifti object.

---

resample_voi	<i>Resample a VOI to match a target image space using nearest-neighbour interpolation.</i>
--------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

volume data as a nifti object.

---

reslice\_to\_mrs                *Reslice a nifti object to match the orientation of mrs data.*

---

**Description**

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

**Usage**

```
reslice_to_mrs(mri, mrs, interp = 3L)
```

**Arguments**

mri                nifti object to be resliced.  
 mrs                mrs\_data object for the target orientation.  
 interp             interpolation parameter, see niftyreg.linear definition.

**Value**

resliced imaging data.

---

reson\_table2mrs\_data        *Generate mrs\_data from a table of single Lorentzian resonances.*

---

**Description**

Generate mrs\_data from a table of single Lorentzian resonances.

**Usage**

```

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

```

**Arguments**

reson\_table as produced by the hsvd function.  
 acq\_paras list of acquisition parameters. See  
 back\_extrap\_pts number of data points to back extrapolate [def\\_acq\\_paras](#)

**Value**

mrs\_data object.

---

re_weighting	<i>Apply a weighting to the FID to enhance spectral resolution.</i>
--------------	---

---

**Description**

Apply a weighting to the FID to enhance spectral resolution.

**Usage**

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

**Arguments**

mrs\_data data to be enhanced.  
 re resolution enhancement factor (rising exponential factor).  
 alpha alpha factor (Gaussian decay)

**Value**

resolution enhanced mrs\_data.

---

rm_dyns	<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	<i>Apply water reference scaling to a fitting results object to yield metabolite quantities in millimolar (mM) units (mol / kg of tissue water).</i>
-----------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

A `fit_result` object with a rescaled results table.

---

<code>scale_amp_molal_pvc</code>	<i>Apply water reference scaling to a fitting results object to yield metabolite quantities in millimolar (mM) units (mol / kg of tissue water).</i>
----------------------------------	--

---

**Description**

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

**Usage**

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

**Arguments**

<code>fit_result</code>	result object generated from fitting.
<code>ref_data</code>	water reference MRS data object.
<code>p_vols</code>	a numeric vector of partial volumes expressed as percentages. For example, a voxel containing 100% white matter tissue would use : <code>p_vols = c(WM = 100, GM = 0, CSF = 0)</code> .
<code>te</code>	the MRS TE in seconds.
<code>tr</code>	the MRS TR in seconds.
<code>...</code>	additional arguments to <code>get_td_amp</code> function.

**Value**

A `fit_result` object with a rescaled results table.

---

scale_amp_molar	<i>Apply water reference scaling to a fitting results object to yield metabolite quantities in millimolar (mM) units (mol / Litre of tissue).</i>
-----------------	---

---

### Description

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

### Usage

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

### Arguments

fit_result	a result object generated from fitting.
ref_data	water reference MRS data object.
w_att	water attenuation factor (default = 0.7). Assumes water T2 of 80ms and a TE = 30 ms. $\exp(-30\text{ms} / 80\text{ms}) \sim 0.7$ .
w_conc	assumed water concentration (default = 35880). Default value corresponds to typical white matter. Set to 43300 for gray matter, and 55556 for phantom measurements.
...	additional arguments to get_td_amp function.

### Value

a fit\_result object with a rescaled results table.

---

scale_amp_molar2molal_pvc	<i>Convert default LCM/TARQUIN concentration scaling to molal units with partial volume correction.</i>
---------------------------	---

---

### Description

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

### Usage

```
scale_amp_molar2molal_pvc(fit_result, p_vols, te, tr)
```

**Arguments**

fit_result	a fit_result object to apply partial volume correction.
p_vols	a numeric vector of partial volumes expressed as percentages. For example, a voxel containing 100% white matter tissue would use : p_vols = c(WM = 100, GM = 0, CSF = 0).
te	the MRS TE.
tr	the MRS TR.

**Value**

a fit\_result object with a rescaled results table.

---

scale_amp_ratio	<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, use_mean_value = FALSE)
```

**Arguments**

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

**Value**

a fit\_result object with a rescaled results table.

---

scale\_amp\_ratio\_value *Scale fitted amplitudes to a ratio of signal amplitude.*

---

**Description**

Scale fitted amplitudes to a ratio of signal amplitude.

**Usage**

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

**Arguments**

fit\_result      a result object generated from fitting.  
value            the number use as a denominator.

**Value**

a fit\_result object with a rescaled results table.

---

scale\_amp\_water\_ratio *Scale metabolite amplitudes as a ratio to the unsuppressed water amplitude.*

---

**Description**

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

**Usage**

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

**Arguments**

fit\_result      a result object generated from fitting.  
ref\_data        a water reference MRS data object.  
...             additional arguments to get\_td\_amp function.

**Value**

a fit\_result object with a rescaled results table.

---

scale_basis_amp	<i>Scale a basis object by a scalar.</i>
-----------------	--

---

**Description**

Scale a basis object by a scalar.

**Usage**

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

**Arguments**

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

**Value**

basis\_set object multiplied by the amplitude scale factor.

---

scale_basis_from_singlet	<i>Scale a basis-set to be consistent with spant assumptions for water scaling.</i>
--------------------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

a scaled basis.

---

scale_mrs_amp	<i>Scale an mrs_data object by a scalar or vector or amplitudes.</i>
---------------	--

---

**Description**

Scale an mrs\_data object by a scalar or vector or amplitudes.

**Usage**

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

**Arguments**

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

**Value**

mrs\_data object multiplied by the amplitude scale factor.

---

scale_spec	<i>Scale mrs_data to a spectral region.</i>
------------	---

---

**Description**

Scale mrs\_data to a spectral region.

**Usage**

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

**Arguments**

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

**Value**

normalised data.

---

<code>sd</code>	<i>Calculate the standard deviation spectrum from an <code>mrs_data</code> object.</i>
-----------------	--

---

**Description**

Calculate the standard deviation spectrum from an `mrs_data` object.

**Usage**

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

**Arguments**

<code>x</code>	object of class <code>mrs_data</code> .
<code>na.rm</code>	remove NA values.

**Value**

sd `mrs_data` object.

---

<code>sd.mrs_data</code>	<i>Calculate the standard deviation spectrum from an <code>mrs_data</code> object.</i>
--------------------------	--

---

**Description**

Calculate the standard deviation spectrum from an `mrs_data` object.

**Usage**

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

**Arguments**

<code>x</code>	object of class <code>mrs_data</code> .
<code>na.rm</code>	remove NA values.

**Value**

`sd` `mrs_data` object.

---

<code>seconds</code>	<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**

<code>mrs_data</code>	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	<i>MEGA-PRESS sequence with ideal localisation pulses and Gaussian shaped editing pulse.</i>
----------------------	--

---

**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	<i>PRESS sequence with ideal pulses.</i>
-----------------	--

---

**Description**

PRESS sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

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

---

**Description**

Simple pulse and acquire sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

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

---

**Description**

Simple pulse and acquire sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

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

---

**Description**

sLASER sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

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

---

**Description**

Spin echo sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

seq\_spin\_echo\_ideal\_31p

*Spin echo sequence with ideal pulses.*

---

**Description**

Spin echo sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

seq\_steam\_ideal

*STEAM sequence with ideal pulses.*

---

**Description**

STEAM sequence with ideal pulses.

**Usage**

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

**Arguments**

spin_params	spin system definition.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.
TE	sequence parameter in seconds.
TM	sequence parameter in seconds.
amp_scale	amplitude scaling factor. Set to 2 (default) to ensure correct scaling for water reference scaling. Set to 1 to maintain the inherent loss of signal associated with STEAM.

**Value**

list of resonance amplitudes and frequencies.

---

seq\_steam\_ideal\_cof     *STEAM sequence with ideal pulses and coherence order filtering to simulate gradient crushers.*

---

**Description**

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

**Usage**

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

**Arguments**

spin_params	spin system definition.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.
TE	sequence parameter in seconds.
TM	sequence parameter in seconds.
amp_scale	amplitude scaling factor. Set to 2 (default) to ensure correct scaling for water reference scaling. Set to 1 to maintain the inherent loss of signal associated with STEAM.

**Value**

list of resonance amplitudes and frequencies.

---

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

---

**Description**

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

**Usage**

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

**Arguments**

spin_params	spin system definition.
ft	transmitter frequency in Hz.
ref	reference value for ppm scale.
TE	sequence parameter in seconds.
TM	sequence parameter in seconds.
amp_scale	amplitude scaling factor. Set to 2 (default) to ensure correct scaling for water reference scaling. Set to 1 to maintain the inherent loss of signal associated with STEAM.

**Value**

list of resonance amplitudes and frequencies.

---

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

**Arguments**

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

---

set_lcm_cmd	<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	path to binary.
-----	-----------------

---

set_lw	<i>Apply line-broadening to an mrs_data object to achieve a specified linewidth.</i>
--------	--

---

**Description**

Apply line-broadening to an mrs\_data object to achieve a specified linewidth.

**Usage**

```
set_lw(mrs_data, lw, xlim = c(4, 0.5), lg = 1, mask_narrow = TRUE)
```

**Arguments**

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

**Value**

line-broadened data.

---

set_mask_xy_mat	<i>Set the masked voxels in a 2D MRSI dataset to given spectrum.</i>
-----------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

updated dataset.

---

set_Ntrans	<i>Set the number of transients for an mrs_data object.</i>
------------	---

---

**Description**

Set the number of transients for an mrs\_data object.

**Usage**

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

**Arguments**

mrs_data	MRS data.
n_trans	number of acquired transients.

---

set_precomp_mode	<i>Set the precompute mode.</i>
------------------	---------------------------------

---

**Description**

Set the precompute mode.

**Usage**

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

**Arguments**

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

---

set_precomp_verbose	<i>Set the verbosity of the precompute function.</i>
---------------------	--

---

**Description**

Set the verbosity of the precompute function.

**Usage**

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

**Arguments**

verbose	can be TRUE or FALSE.
---------	-----------------------

---

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.
-----	-----------------

---

set_tr	<i>Set the repetition time of an MRS dataset.</i>
--------	---

---

**Description**

Set the repetition time of an MRS dataset.

**Usage**

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

**Arguments**

mrs_data	MRS data.
tr	repetition time in seconds.

**Value**

updated mrs\_data set.

---

shift	<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.

---

shift_basis	<i>Apply frequency shifts to basis set signals.</i>
-------------	---

---

**Description**

Apply frequency shifts to basis set signals.

**Usage**

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

**Arguments**

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

**Value**

modified basis set object.

---

sim_basis	<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\_mm\_lip\_lcm *Simulate a macromolecular and lipid basis-set suitable for 1H brain MRS analysis.*

---

### Description

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

### Usage

```
sim_basis_mm_lip_lcm(acq_params = def_acq_params())
```

**Arguments**

acq\_paras      list of acquisition parameters or an mrs\_data object. See [def\\_acq\\_paras](#)

**Value**

basis object.

---

sim_basis_tqn	<i>Simulate a basis file using TARQUIN.</i>
---------------	---

---

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

### Arguments

acq_paras	list of acquisition parameters or an <code>mrs_data</code> 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.
basis_lb	apply additional Gaussian line-broadening to the basis (Hz).
zero_lip_mm	zero the amplitudes of any lipid or macromolecular components based on their name starting with "MM" or "Lip".
remove_lip_mm	remove any lipid or macromolecular basis components based on their name starting with "MM" or "Lip".
...	extra parameters to pass to the pulse sequence function.

### Value

see `full_output` option.

---

sim_mol	<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 an mrs_data object containing simulated Gaussian noise.</i>
-----------	---

---

### Description

Simulate an mrs\_data object containing simulated Gaussian noise.

### Usage

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

### Arguments

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

### Value

mrs\_data object.

---

sim_resonances	<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(),  
  fp_scale = TRUE,  
  back_extrap_pts = 0,  
  sum_resonances = TRUE  
)
```

**Arguments**

freq	resonance frequency.
amp	resonance amplitude.
lw	line width in Hz.
lg	Lorentz-Gauss lineshape parameter (between 0 and 1).
phase	phase in degrees.
freq_ppm	frequencies are given in ppm units if set to TRUE, otherwise Hz are assumed.
acq_paras	list of acquisition parameters. See <a href="#">def_acq_paras</a>
fp_scale	multiply the first data point by 0.5.
back_extrap_pts	number of data points to back extrapolate.
sum_resonances	sum all resonances (default is TRUE), otherwise return a dynamic mrs_data object.

**Value**

MRS data object.

**Examples**

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

---

sim\_th\_excit\_profile    *Simulate an ideal pulse excitation profile by smoothing a top-hat function with a Gaussian.*

---

### Description

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

### Usage

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

### Arguments

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

### Value

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

---

sim\_zero                    *Simulate an mrs\_data object containing complex zero valued samples.*

---

### Description

Simulate an mrs\_data object containing complex zero valued samples.

### Usage

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

### Arguments

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

### Value

mrs\_data object.

---

smooth_dyns	<i>Smooth data across the dynamic dimension with a Gaussian kernel.</i>
-------------	---

---

**Description**

Smooth data across the dynamic dimension with a Gaussian kernel.

**Usage**

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

**Arguments**

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

**Value**

smoothed mrs\_data object.

---

sort_basis	<i>Sort the basis-set elements alphabetically.</i>
------------	--

---

**Description**

Sort the basis-set elements alphabetically.

**Usage**

```
sort_basis(basis)
```

**Arguments**

basis	input basis.
-------	--------------

**Value**

sorted basis.

---

spant\_abfit\_benchmark *Simulate and fit some spectra with ABfit for benchmarking purposes. Basic timing and performance metrics will be printed.*

---

**Description**

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

**Usage**

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

**Arguments**

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

---

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

---

**Description**

Example MEGA-PRESS data with significant B0 drift.

**Usage**

```
spant_mpress_drift
```

**Format**

An object of class mrs\_data of length 13.

---

spant\_simulation\_benchmark

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

---

### Description

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

### Usage

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

### Arguments

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

---

spec\_decomp

*Decompose an mrs\_data object into white and gray matter spectra.*

---

### Description

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

### Usage

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

### Arguments

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

### Value

a list of two mrs\_data objects corresponding to the two tissue types.

---

spec_op	<i>Perform a mathematical operation on a spectral region.</i>
---------	---

---

**Description**

Perform a mathematical operation on a spectral region.

**Usage**

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

**Arguments**

mrs_data	MRS data.
xlim	spectral range to be integrated (defaults to full range).
operator	can be "sum" (default), "mean", "I2", "max", "min" or "max-min".
freq_scale	units of xlim, can be : "ppm", "hz" or "points".
mode	spectral mode, can be : "re", "im", "mod" or "cplx".

**Value**

an array of integral values.

---

spin_sys	<i>Create a spin system object for pulse sequence simulation.</i>
----------	---

---

**Description**

Create a spin system object for pulse sequence simulation.

**Usage**

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

**Arguments**

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

**Value**

spin system object.

---

spm_pve2categorical	<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.

---

ssp	<i>Signal space projection method for lipid suppression.</i>
-----	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

lipid suppressed mrs\_data object.

---

stackplot	<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	<i>Plot the fitting results of an object of class fit_result with individual basis set components shown.</i>
----------------------	--

---

**Description**

Plot the fitting results of an object of class fit\_result with individual basis set components shown.

**Usage**

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

```

    mar = NULL,
    show_grid = TRUE,
    grid_nx = NULL,
    grid_ny = NA,
    ...
)

```

### Arguments

x	fit_result object.
xlim	the range of values to display on the x-axis, eg xlim = c(4,1).
y_offset	separate basis signals in the y-axis direction by this value.
dyn	the dynamic index to plot.
x_pos	the x index to plot.
y_pos	the y index to plot.
z_pos	the z index to plot.
coil	the coil element number to plot.
n	single index element to plot (overrides other indices when given).
sub_bl	subtract the baseline from the data and fit (logical).
labels	print signal labels at the right side of the plot.
label_names	provide a character vector of signal names to replace the defaults determined from the basis set.
sig_col	colour of individual signal components.
restore_def_par	restore default plotting par values after the plot has been made.
omit_signals	a character vector of basis signal names to be removed from the plot.
combine_lipmm	combine all basis signals with names starting with "Lip" or "MM".
combine_metab	combine all basis signals with names not starting with "Lip" or "MM".
mar	option to adjust the plot margins. See ?par.
show_grid	plot gridlines behind the data (logical). Defaults to TRUE.
grid_nx	number of cells of the grid in x and y direction. When NULL the grid aligns with the tick marks on the corresponding default axis (i.e., tickmarks as computed by axTicks). When NA, no grid lines are drawn in the corresponding direction.
grid_ny	as above.
...	further arguments to plot method.

---

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

---

## Description

Stackplot plotting method for objects of class mrs\_data.

## Usage

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

## Arguments

x	object of class mrs_data.
xlim	the range of values to display on the x-axis, eg xlim = c(4,1).
mode	representation of the complex numbers to be plotted, can be one of: "re", "im", "mod" or "arg".

x_units	the units to use for the x-axis, can be one of: "ppm", "hz", "points" or "seconds".
fd	display data in the frequency-domain (default), or time-domain (logical).
col	set the colour of the line, eg col = rgb(1, 0, 0, 0.5).
alpha	set the line transparency, eg alpha = 0.5 is 50% transparency. Overrides any transparency levels set by col.
x_offset	separate plots in the x-axis direction by this value. Default value is 0.
y_offset	separate plots in the y-axis direction by this value.
plot_dim	the dimension to display on the y-axis, can be one of: "dyn", "x", "y", "z", "coil" or NULL. If NULL (the default) all spectra will be collapsed into the dynamic dimension and displayed.
x_pos	the x index to plot.
y_pos	the y index to plot.
z_pos	the z index to plot.
dyn	the dynamic index to plot.
coil	the coil element number to plot.
bty	option to draw a box around the plot. See ?par.
labels	add labels to each data item.
lab_cex	label size.
right_marg	change the size of the right plot margin.
bl_lty	linetype for the y = 0 baseline trace. A default value NULL results in no baseline being plotted.
restore_def_par	restore default plotting par values after the plot has been made.
show_grid	plot gridlines behind the data (logical). Defaults to TRUE.
grid_nx	number of cells of the grid in x and y direction. When NULL the grid aligns with the tick marks on the corresponding default axis (i.e., tickmarks as computed by axTicks). When NA, no grid lines are drawn in the corresponding direction.
grid_ny	as above.
lwd	plot linewidth.
vline	x-value to draw a vertical line.
...	other arguments to pass to the matplot method.

---

sub_first_dyn	<i>Subtract the first dynamic spectrum from a dynamic series.</i>
---------------	---

---

**Description**

Subtract the first dynamic spectrum from a dynamic series.

**Usage**

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

**Arguments**

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

**Value**

subtracted data.

---

sub_mean_dyns	<i>Subtract the mean dynamic spectrum from a dynamic series.</i>
---------------	--

---

**Description**

Subtract the mean dynamic spectrum from a dynamic series.

**Usage**

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

**Arguments**

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

**Value**

subtracted data.

---

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.

---

sum_mrs	<i>Sum two mrs_data objects.</i>
---------	----------------------------------

---

**Description**

Sum two mrs\_data objects.

**Usage**

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

**Arguments**

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

**Value**

a + b

---

sum_mrs_list	<i>Return the sum of a list of mrs_data objects.</i>
--------------	--

---

**Description**

Return the sum of a list of mrs\_data objects.

**Usage**

```
sum_mrs_list(mrs_list)
```

**Arguments**

mrs_list	list of mrs_data objects.
----------	---------------------------

**Value**

sum mrs\_data object.

---

svs\_1h\_brain\_analysis *Standard SVS 1H brain analysis pipeline.*

---

### Description

Standard SVS 1H brain analysis pipeline.

### Usage

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

### Arguments

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

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

**Value**

a fit\_result or mrs\_data object depending on the preproc\_only option.

---

svs\_1h\_brain\_batch\_analysis

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

a list of fit\_result objects.

---

td2fd	<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.

---

tr	<i>Return the repetition time of an MRS dataset.</i>
----	--

---

**Description**

Return the repetition time of an MRS dataset.

**Usage**

```
tr(mrs_data)
```

**Arguments**

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

**Value**

repetition time in seconds.

---

varpro\_3\_para\_opts      *Return a list of options for VARPRO based fitting with 3 free parameters.*

---

### Description

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

### Usage

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

### Arguments

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

### Value

list of options.

---

varpro\_basic\_opts      *Return a list of options for a basic VARPRO analysis.*

---

### Description

Return a list of options for a basic VARPRO analysis.

### Usage

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

**Arguments**

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

**Value**

full list of options.

---

varpro_opts	<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(),  
  nuc = def_nuc(),  
  dyns = 1,  
  fd = FALSE  
)
```

**Arguments**

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

**Value**

mrs\_data object.

---

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

---

**Description**

Write a basis object to an LCMModel .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 LCMModel.

---

write_basis_tqn	<i>Generate a basis file using TARQUIN.</i>
-----------------	---

---

**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	<i>Write MRS data object to file.</i>
-----------	---------------------------------------

---

**Description**

Write MRS data object to file.

**Usage**

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

**Arguments**

mrs_data	object to be written to file, or list of mrs_data objects.
fname	one or more filenames to output.
format	string describing the data format. Must be one of the following : "nifti", "dpt", "lcm_raw", "rds". If not specified, the format will be guessed from the filename extension.
force	set to TRUE to overwrite any existing files.

---

write_mrs_nifti	<i>Write MRS data object to file in NIFTI format.</i>
-----------------	---

---

**Description**

Write MRS data object to file in NIFTI format.

**Usage**

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

**Arguments**

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

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

modified mrs\_data object.

---

zero_higher_orders	<i>Zero all coherences including and above a given order.</i>
--------------------	---

---

**Description**

Zero all coherences including and above a given order.

**Usage**

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

**Arguments**

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

**Value**

density matrix.

---

zero_td_pts_end	<i>Set mrs_data object data points at the end of the FID to zero.</i>
-----------------	---

---

**Description**

Set mrs\_data object data points at the end of the FID to zero.

**Usage**

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

**Arguments**

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

**Value**

modified mrs\_data object.

---

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

---

**Description**

Zero-fill MRS data in the time domain.

**Usage**

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

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

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

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

**Arguments**

x	input mrs_data or basis_set object.
factor	zero-filling factor, factor of 2 returns a dataset with twice the original data points.
offset	number of points from the end of the FID to insert the zero values.

**Value**

zero-filled data.

---

zf\_xy

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

zero-filled data.

# Index

## \* datasets

spant\_mpress\_drift, 173

abfit\_opts, 9, 13

abfit\_opts\_v1\_9\_0, 13

acquire, 13

add\_noise, 14

align, 14

apodise\_xy, 15

append\_basis, 16

append\_coils, 16

append\_dyns, 17

apply\_axes, 17

apply\_mrs, 18

apply\_pulse, 18

Arg.mrs\_data, 19

array2mrs\_data, 19

auto\_phase, 20

back\_extrap\_ar, 21

basis2mrs\_data, 21

bbase, 22

bc\_als, 23

bc\_constant, 23

bc\_gauss, 24

bc\_poly, 24

bc\_spline, 25

beta2lw, 25

bin\_spec, 26

calc\_coil\_noise\_cor, 26

calc\_coil\_noise\_sd, 27

calc\_ed\_from\_lambda, 27

calc\_peak\_info\_vec, 28

calc\_sd\_poly, 28

calc\_spec\_diff, 29

calc\_spec\_snr, 29

check\_lcm, 30

check\_tqn, 30

circ\_mask, 31

coherence\_filter, 31

collapse\_to\_dyns, 32

comb\_coils, 32

comb\_fit\_list\_fit\_tables, 33

comb\_fit\_list\_result\_tables, 34

comb\_fit\_tables, 35

comb\_metab\_ref, 35

Conj.mrs\_data, 36

conv\_mrs, 36

crop\_basis, 37

crop\_spec, 37

crop\_td\_pts, 38

crop\_td\_pts\_end, 38

crop\_td\_pts\_pot, 39

crop\_xy, 39

crossprod\_3d, 40

decimate\_mrs\_fd, 40

decimate\_mrs\_td, 41

deconv\_mrs, 41

def\_acq\_paras, 42, 141, 163–167, 170

def\_fs, 43

def\_ft, 43

def\_N, 43

def\_nuc, 44

def\_ref, 44

dicom\_reader, 44

diff\_mrs, 45

downsample\_mrs\_fd, 46

downsample\_mrs\_td, 46

ecc, 47

elliptical\_mask, 47

est\_noise\_sd, 48

fd2td, 48

fd\_conv\_filt, 49

fd\_gauss\_smo, 49

find\_mrs\_files, 50

fit\_amps, 50

fit\_diags, 51  
fit\_mrs, 51  
fit\_res2csv, 53  
fit\_t1\_ti\_array, 53  
fit\_t1\_tr\_array, 54  
fit\_t2\_te\_array, 55  
fp\_phase, 55  
fp\_phase\_correct, 56  
fp\_scale, 56  
fs, 57  
ft\_dyns, 57  
ft\_shift, 58  
ft\_shift\_mat, 58  
  
gausswin\_2d, 59  
gen\_F, 59  
gen\_F\_xy, 60  
gen\_I, 60  
get\_1h\_brain\_basis\_paras, 61  
get\_1h\_brain\_basis\_paras\_v1, 61  
get\_1h\_brain\_basis\_paras\_v2, 62  
get\_1h\_brain\_basis\_paras\_v3, 62  
get\_2d\_psf, 63  
get\_acq\_paras, 63  
get\_basis\_subset, 64  
get\_dyns, 64  
get\_even\_dyns, 65  
get\_fh\_dyns, 65  
get\_fit\_map, 66  
get\_fp, 66  
get\_guassian\_pulse, 67  
get\_head\_dyns, 67  
get\_lcm\_cmd, 68  
get\_metab, 68  
get\_mol\_names, 68  
get\_mol\_paras, 69  
get\_mrs\_affine, 71  
get\_mrsi2d\_seg, 69  
get\_mrsi\_voi, 70  
get\_mrsi\_voxel, 70  
get\_mrsi\_voxel\_xy\_psf, 71  
get\_odd\_dyns, 72  
get\_ref, 72  
get\_seg\_ind, 73  
get\_sh\_dyns, 73  
get\_slice, 74  
get\_subset, 74  
get\_svs\_voi, 75  
get\_tail\_dyns, 76  
  
get\_td\_amp, 76  
get\_tqn\_cmd, 77  
get\_uncoupled\_mol, 77  
get\_voi\_cog, 78  
get\_voi\_seg, 78  
get\_voi\_seg\_psf, 79  
get\_voxel, 79  
grid\_shift\_xy, 81  
gridplot, 80  
gridplot.mrs\_data, 80  
  
hsvd, 81  
hsvd\_filt, 82  
hsvd\_vec, 83  
hz, 83  
  
ift\_shift, 84  
ift\_shift\_mat, 84  
Im.mrs\_data, 85  
image.mrs\_data, 85  
img2kspace\_xy, 86  
Imzap, 87  
int\_spec, 88  
interleave\_dyns, 87  
inv\_even\_dyns, 88  
inv\_odd\_dyns, 89  
is.def, 89  
is\_fd, 90  
  
kspace2img\_xy, 90  
  
l2\_reg, 91  
lb, 92  
lofdc, 92  
lw2alpha, 93  
lw2beta, 94  
  
make\_basis\_from\_raw, 94  
mask\_dyns, 95  
mask\_fit\_res, 95  
mask\_xy, 96  
mask\_xy\_corners, 96  
mask\_xy\_ellipse, 97  
mask\_xy\_mat, 97  
mat2mrs\_data, 98  
matexp, 98  
max\_mrs, 99  
max\_mrs\_interp, 99  
mean.list, 100

mean.mrs\_data, 100  
 mean\_dyn\_blocks, 101  
 mean\_dyn\_pairs, 102  
 mean\_dyns, 101  
 mean\_mrs\_list, 102  
 mean\_vec\_blocks, 103  
 median\_dyns, 103  
 Mod.mrs\_data, 104  
 mod\_td, 104  
 mrs\_data2basis, 105  
 mrs\_data2mat, 105  
 mrs\_data2vec, 106  
 mvfftshift, 106  
 mvifftshift, 107  
  
 n2coord, 107  
 Ncoils, 108  
 Ndyns, 108  
 nifti\_flip\_lr, 108  
 Npts, 109  
 Nspec, 109  
 Ntrans, 110  
 Nx, 110  
 Ny, 110  
 Nz, 111  
  
 one\_page\_pdf, 111  
 ortho3, 112  
 ortho3\_inter, 113  
  
 peak\_info, 113  
 pg\_extrap\_xy, 114  
 phase, 115  
 plot\_fit\_result, 115  
 plot\_mrs\_data, 117  
 plot\_bc, 119  
 plot\_slice\_fit, 119  
 plot\_slice\_fit\_inter, 120  
 plot\_slice\_map, 120  
 plot\_slice\_map\_inter, 121  
 plot\_spec\_sd, 123  
 plot\_voi\_overlay, 123  
 plot\_voi\_overlay\_seg, 124  
 ppm, 124  
 precomp, 125  
 print\_fit\_result, 125  
 print\_mrs\_data, 126  
  
 qn\_states, 126  
  
 rats, 127  
 Re.mrs\_data, 128  
 re\_weighting, 141  
 read\_basis, 129  
 read\_ima\_coil\_dir, 129  
 read\_ima\_dyn\_dir, 130  
 read\_lcm\_coord, 130  
 read\_mrs, 131  
 read\_mrs\_tqn, 132  
 read\_pulse\_pta, 133  
 read\_siemens\_txt\_hdr, 133  
 read\_tqn\_fit, 134  
 read\_tqn\_result, 134  
 recon\_imag, 135  
 recon\_imag\_vec, 135  
 recon\_twix\_2d\_mrsi, 136  
 rectangular\_mask, 136  
 rep\_array\_dim, 137  
 rep\_dyn, 137  
 rep\_mrs, 138  
 resample\_basis, 138  
 resample\_img, 139  
 resample\_voi, 139  
 reslice\_to\_mrs, 140  
 reson\_table2mrs\_data, 140  
 rm\_dyns, 141  
  
 scale\_amp\_molal, 142  
 scale\_amp\_molal\_pvc, 143  
 scale\_amp\_molar, 144  
 scale\_amp\_molar2molal\_pvc, 144  
 scale\_amp\_ratio, 145  
 scale\_amp\_ratio\_value, 146  
 scale\_amp\_water\_ratio, 146  
 scale\_basis\_amp, 147  
 scale\_basis\_from\_singlet, 147  
 scale\_mrs\_amp, 148  
 scale\_spec, 148  
 sd, 149  
 sd.mrs\_data, 150  
 seconds, 150  
 seq\_cpmg\_ideal, 151  
 seq\_mega\_press\_ideal, 151  
 seq\_press\_ideal, 152  
 seq\_pulse\_acquire, 153  
 seq\_pulse\_acquire\_31p, 153  
 seq\_slaser\_ideal, 154  
 seq\_spin\_echo\_ideal, 154  
 seq\_spin\_echo\_ideal\_31p, 155

seq\_steam\_ideal, 155  
seq\_steam\_ideal\_cof, 156  
seq\_steam\_ideal\_young, 156  
set\_def\_acq\_paras, 157  
set\_lcm\_cmd, 158  
set\_lw, 158  
set\_mask\_xy\_mat, 159  
set\_Ntrans, 159  
set\_precomp\_mode, 160  
set\_precomp\_verbose, 160  
set\_ref, 160  
set\_td\_pts, 161  
set\_tqn\_cmd, 161  
set\_tr, 162  
shift, 162  
shift\_basis, 163  
sim\_basis, 163  
sim\_basis\_1h\_brain, 164  
sim\_basis\_1h\_brain\_press, 165  
sim\_basis\_mm\_lip\_lcm, 165  
sim\_basis\_tqn, 166  
sim\_brain\_1h, 167  
sim\_mol, 168  
sim\_noise, 169  
sim\_resonances, 169  
sim\_th\_excit\_profile, 171  
sim\_zero, 171  
smooth\_dyns, 172  
sort\_basis, 172  
spant (spant-package), 8  
spant-package, 8  
spant\_abfit\_benchmark, 173  
spant\_mpress\_drift, 173  
spant\_simulation\_benchmark, 174  
spec\_decomp, 174  
spec\_op, 175  
spin\_sys, 175  
spm\_pve2categorical, 176  
ssp, 176  
stackplot, 177  
stackplot.fit\_result, 177  
stackplot.mrs\_data, 179  
sub\_first\_dyn, 181  
sub\_mean\_dyns, 181  
sum\_coils, 182  
sum\_dyns, 182  
sum\_mrs, 183  
sum\_mrs\_list, 183  
svs\_1h\_brain\_analysis, 184  
svs\_1h\_brain\_batch\_analysis, 185  
td2fd, 186  
td\_conv\_filt, 187  
tdsr, 186  
tr, 187  
varpro\_3\_para\_opts, 188  
varpro\_basic\_opts, 188  
varpro\_opts, 189  
vec2mrs\_data, 190  
write\_basis, 191  
write\_basis\_tqn, 191  
write\_mrs, 192  
write\_mrs\_nifti, 192  
zero\_fade\_spec, 193  
zero\_higher\_orders, 193  
zero\_td\_pts\_end, 194  
zf, 194  
zf\_xy, 195