

# Package ‘spant’

May 8, 2026

**Type** Package

**Title** MR Spectroscopy Analysis Tools

**Version** 4.0.0

**Date** 2026-05-08

**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)>, Wilson (2025) <[DOI:10.1002/mrm.30462](https://doi.org/10.1002/mrm.30462)> 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

**NeedsCompilation** yes

**Depends** R (>= 4.1.0)

**Imports** abind, plyr, pbapply, pracma, stringr, expm (>= 1.0-0), signal, minpack.lm, utils, graphics, grDevices, ptw, mmand, RNifti, RNiftyReg, fields, numDeriv, nloptr, irlba, jsonlite

**Suggests** viridisLite, shiny, shinyFiles, ggplot2, miniUI, knitr, kableExtra, rmarkdown, testthat, ragg, doParallel, parallel, digest, readxl, fslr, car, divest, rpyANTs

**VignetteBuilder** knitr

**Encoding** UTF-8

**Language** en-GB

**URL** <https://spantdoc.wilsonlab.co.uk/>,  
<https://martin3141.github.io/spant/>,  
<https://github.com/martin3141/spant/>

**Config/roxygen2/version** 8.0.0

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

**Maintainer** Martin Wilson <[martin.wilson111@gmail.com](mailto:martin.wilson111@gmail.com)>

**Repository** CRAN

**Date/Publication** 2026-05-08 11:40:02 UTC

## Contents

spant-package . . . . .	10
abfit_opts . . . . .	11
abfit_opts_v1_9_0 . . . . .	15
abfit_reg_opts . . . . .	15
acquire . . . . .	19
add_noise . . . . .	20
add_noise_spec_snr . . . . .	20
align . . . . .	21
apodise_xy . . . . .	22
append_basis . . . . .	22
append_coils . . . . .	23
append_dyns . . . . .	23
append_regs . . . . .	24
apply_axes . . . . .	24
apply_mrs . . . . .	25
apply_pulse . . . . .	25
Arg.mrs_data . . . . .	26
array2mrs_data . . . . .	26
assign_dyns . . . . .	27
auto_phase . . . . .	28
auto_phase_bl . . . . .	28
back_extrap_ar . . . . .	29
basis2dyn_mrs_data . . . . .	30
basis2mrs_data . . . . .	30
bbase . . . . .	31
bc_als . . . . .	32
bc_constant . . . . .	32
bc_gauss . . . . .	33
bc_poly . . . . .	33
bc_spline . . . . .	34
beta2lw . . . . .	34
bin_spec . . . . .	35
calc_basis_corr_mat . . . . .	35
calc_basis_crlbs . . . . .	36
calc_coil_noise_cor . . . . .	36
calc_coil_noise_sd . . . . .	37
calc_design_efficiency . . . . .	37
calc_ed_from_lambda . . . . .	38
calc_peak_info_vec . . . . .	38
calc_sd_poly . . . . .	39
calc_spec_diff . . . . .	39
calc_spec_snr . . . . .	40

check_lcm . . . . .	41
check_tqn . . . . .	41
circ_mask . . . . .	41
coherence_filter . . . . .	42
collapse_to_dyns . . . . .	42
comb_coils . . . . .	43
comb_coils_mrsi_gls . . . . .	44
comb_coils_svs_gls . . . . .	44
comb_fit_list_dyns . . . . .	45
comb_fit_list_fit_tables . . . . .	45
comb_fit_list_result_tables . . . . .	46
comb_fit_tables . . . . .	47
comb_metab_ref . . . . .	47
Conj.mrs_data . . . . .	48
conv_mrs . . . . .	48
crop_basis . . . . .	49
crop_spec . . . . .	49
crop_td_pts . . . . .	50
crop_td_pts_end . . . . .	50
crop_td_pts_pot . . . . .	51
crop_xy . . . . .	51
crossprod_3d . . . . .	52
decimate_mrs_fd . . . . .	52
decimate_mrs_td . . . . .	53
deconv_mrs . . . . .	53
def_acq_paras . . . . .	54
def_fs . . . . .	55
def_ft . . . . .	55
def_N . . . . .	55
def_nuc . . . . .	56
def_ref . . . . .	56
dicom_reader . . . . .	56
diff_mrs . . . . .	57
downsample_mrs_fd . . . . .	58
downsample_mrs_td . . . . .	58
dyn_acq_times . . . . .	59
ecc . . . . .	59
elliptical_mask . . . . .	60
est_noise_sd . . . . .	61
faceoff . . . . .	61
fd2td . . . . .	62
fd_conv_filt . . . . .	62
fd_gauss_smo . . . . .	63
find_bids_mrs . . . . .	63
find_mrs_files . . . . .	64
fit_amps . . . . .	64
fit_asy_pvoigt . . . . .	65
fit_diags . . . . .	65

fit_mrs . . . . .	66
fit_res2csv . . . . .	67
fit_svs . . . . .	68
fit_svs_edited . . . . .	72
fit_svs_edited_group_results . . . . .	76
fit_svs_group_results . . . . .	76
fit_svs_gui . . . . .	77
fit_t1_ti_array . . . . .	77
fit_t1_tr_array . . . . .	78
fit_t2_te_array . . . . .	79
fp_phase . . . . .	79
fp_phase_correct . . . . .	80
fp_scale . . . . .	80
fs . . . . .	81
ft_dyns . . . . .	81
ft_shift . . . . .	82
ft_shift_mat . . . . .	82
gausswin_2d . . . . .	83
gen_baseline_reg . . . . .	83
gen_bold_reg . . . . .	84
gen_conv_reg . . . . .	85
gen_F . . . . .	86
gen_F_xy . . . . .	86
gen_group_reg . . . . .	87
gen_I . . . . .	87
gen_impulse_reg . . . . .	88
gen_numeric_reg . . . . .	88
gen_poly_reg . . . . .	89
gen_trap_reg . . . . .	90
get_1h_braino_basis_names . . . . .	91
get_1h_brain_basis_names . . . . .	91
get_1h_brain_basis_paras . . . . .	92
get_1h_brain_basis_paras_v1 . . . . .	92
get_1h_brain_basis_paras_v2 . . . . .	93
get_1h_brain_basis_paras_v3 . . . . .	93
get_1h_spectre_basis_names . . . . .	94
get_2d_psf . . . . .	94
get_acq_paras . . . . .	95
get_ants_dir . . . . .	95
get_basis_subset . . . . .	96
get_dyns . . . . .	96
get_even_dyns . . . . .	97
get_fh_dyns . . . . .	97
get_fit_map . . . . .	98
get_fp . . . . .	98
get_guassian_pulse . . . . .	99
get_head_dyns . . . . .	99
get_hrf . . . . .	100

get_lcm_cmd . . . . .	100
get_metab . . . . .	100
get_mol_names . . . . .	101
get_mol_paras . . . . .	101
get_mrsi2d_seg . . . . .	102
get_mrsi_voi . . . . .	102
get_mrsi_voxel . . . . .	103
get_mrsi_voxel_xy_psf . . . . .	103
get_mrs_affine . . . . .	104
get_odd_dyns . . . . .	104
get_ref . . . . .	105
get_seg_ind . . . . .	105
get_sh_dyns . . . . .	106
get_slice . . . . .	106
get_spant_resources_dir . . . . .	107
get_spin_num . . . . .	107
get_subset . . . . .	108
get_svs_voi . . . . .	109
get_tail_dyns . . . . .	109
get_td_amp . . . . .	110
get_tqn_cmd . . . . .	110
get_uncoupled_mol . . . . .	111
get_voi_cog . . . . .	111
get_voi_seg . . . . .	112
get_voi_seg_psf . . . . .	112
get_voxel . . . . .	113
glm_spec . . . . .	113
glm_spec_fmrs_fl . . . . .	114
glm_spec_fmrs_group . . . . .	115
glm_spec_group_linhyp . . . . .	115
gridplot . . . . .	116
gridplot.mrs_data . . . . .	116
grid_shift_xy . . . . .	117
hsvd . . . . .	117
hsvd_filt . . . . .	118
hsvd_vec . . . . .	119
hz . . . . .	119
ift_shift . . . . .	120
ift_shift_mat . . . . .	120
Im.mrs_data . . . . .	121
image.mrs_data . . . . .	121
img2kspace_xy . . . . .	123
Imzap . . . . .	123
install_ants . . . . .	124
install_cli . . . . .	124
install_faceoff . . . . .	124
install_oasis_template . . . . .	125
interleave_dyns . . . . .	125

int_spec . . . . .	126
inv_even_dyns . . . . .	126
inv_odd_dyns . . . . .	127
is.def . . . . .	127
is_fd . . . . .	128
kspace2img_xy . . . . .	128
l2_reg . . . . .	129
lb . . . . .	130
lb_renoise . . . . .	130
lofdc . . . . .	131
lw2alpha . . . . .	132
lw2beta . . . . .	133
make_basis_from_raw . . . . .	133
mask_dyns . . . . .	134
mask_fit_res . . . . .	134
mask_xy . . . . .	135
mask_xy_corners . . . . .	135
mask_xy_ellipse . . . . .	136
mask_xy_mat . . . . .	136
mat2mrs_data . . . . .	137
match_files . . . . .	137
match_lineshape . . . . .	138
matexp . . . . .	139
max_mrs . . . . .	140
max_mrs_interp . . . . .	140
mean.list . . . . .	141
mean.mrs_data . . . . .	141
mean_dyns . . . . .	142
mean_dyns_scheme . . . . .	142
mean_dyn_blocks . . . . .	143
mean_dyn_pairs . . . . .	143
mean_mrs_list . . . . .	144
mean_vec_blocks . . . . .	144
median_dyns . . . . .	145
Mod.mrs_data . . . . .	145
mod_td . . . . .	146
mrs_data2basis . . . . .	146
mrs_data2bids . . . . .	147
mrs_data2list . . . . .	148
mrs_data2mat . . . . .	148
mrs_data2spec_mat . . . . .	149
mrs_data2vec . . . . .	149
mr_data2bids . . . . .	150
mvfftshift . . . . .	151
mvifftshift . . . . .	151
n2coord . . . . .	152
Ncoils . . . . .	152
Ndyns . . . . .	152

nifti_flip_lr	153
Npts	153
Nspec	154
Ntrans	154
Nx	154
Ny	155
Nz	155
one_page_pdf	155
ortho3	156
ortho3_inter	157
paths2df	158
peak_info	158
pg_extrap_xy	159
phase	160
phase_ref_1h_brain	160
plot.fit_result	161
plot.mrs_data	162
plot_bc	165
plot_reg	165
plot_slice_fit	166
plot_slice_fit_inter	166
plot_slice_map	167
plot_slice_map_inter	168
plot_spec_sd	169
plot_voi_overlay	169
plot_voi_overlay_seg	170
ppm	170
precomp	171
preproc_svs	171
preproc_svs_dataset	172
print.fit_result	173
print.mrs_data	173
qn_states	174
rats	174
Re.mrs_data	176
read_basis	176
read_basis_niidir	177
read_dkd_moco_log	177
read_ima_coil_dir	178
read_ima_dyn_dir	178
read_lcm_coord	179
read_mrs	179
read_mrs_tqn	181
read_pulse_ascii	182
read_pulse_bruker	182
read_pulse_pta	183
read_siemens_txt_hdr	183
read_tqn_fit	184

read_tqn_result . . . . .	184
recon_imag . . . . .	185
recon_imag_vec . . . . .	185
recon_twix_2d_mrsi . . . . .	186
rectangular_mask . . . . .	186
rep_array_dim . . . . .	187
rep_dyn . . . . .	187
rep_mrs . . . . .	188
resample_basis . . . . .	188
resample_img . . . . .	189
resample_voi . . . . .	189
reslice_to_mrs . . . . .	190
reson_table2mrs_data . . . . .	190
re_weighting . . . . .	191
rm_dyns . . . . .	191
scale_amp_legacy . . . . .	192
scale_amp_molal . . . . .	193
scale_amp_molal_pvc . . . . .	194
scale_amp_molar . . . . .	194
scale_amp_molar2molal_pvc . . . . .	195
scale_amp_ratio . . . . .	196
scale_amp_ratio_value . . . . .	196
scale_amp_water_ratio . . . . .	197
scale_basis_amp . . . . .	197
scale_basis_from_singlet . . . . .	198
scale_mrs_amp . . . . .	198
scale_spec . . . . .	199
sd.list . . . . .	199
sd.mrs_data . . . . .	200
seconds . . . . .	201
segment_t1_ants . . . . .	201
segment_t1_fsl . . . . .	202
segment_t1_rpyants . . . . .	202
seq_cpmg_ideal . . . . .	203
seq_mega_press_ideal . . . . .	203
seq_press_2d_shaped . . . . .	204
seq_press_ideal . . . . .	205
seq_pulse_acquire . . . . .	206
seq_slaser_ideal . . . . .	206
seq_spin_echo_ideal . . . . .	207
seq_steam_ideal . . . . .	207
seq_steam_ideal_cof . . . . .	208
seq_steam_ideal_young . . . . .	208
set_ants_dir . . . . .	209
set_def_acq_paras . . . . .	210
set_lcm_cmd . . . . .	210
set_lw . . . . .	211
set_mask_xy_mat . . . . .	211

set_Ntrans . . . . .	212
set_precomp_mode . . . . .	212
set_precomp_verbose . . . . .	212
set_ref . . . . .	213
set_td_pts . . . . .	213
set_tqn_cmd . . . . .	214
set_tr . . . . .	214
shift . . . . .	215
shift_basis . . . . .	215
sim_asy_pvoigt . . . . .	216
sim_basis . . . . .	216
sim_basis_1h_brain . . . . .	217
sim_basis_1h_brain_press . . . . .	218
sim_basis_mm_lip_lcm . . . . .	219
sim_basis_tqn . . . . .	219
sim_brain_1h . . . . .	220
sim_mol . . . . .	221
sim_noise . . . . .	222
sim_resonances . . . . .	223
sim_th_excit_profile . . . . .	224
sim_zero . . . . .	224
smooth_dyns . . . . .	225
sort_basis . . . . .	226
spant_abfit_benchmark . . . . .	226
spant_simulation_benchmark . . . . .	227
spant_sim_fmrs_dataset . . . . .	227
spec_decomp . . . . .	228
spec_op . . . . .	228
spin_sys . . . . .	229
spm_pve2categorical . . . . .	230
ssp . . . . .	230
stackplot . . . . .	231
stackplot.fit_result . . . . .	231
stackplot.mrs_data . . . . .	233
subtract_rest_task . . . . .	235
sub_even_from_odd_dyns . . . . .	235
sub_first_dyn . . . . .	236
sub_mean_dyns . . . . .	236
sub_median_dyns . . . . .	237
sub_odd_from_even_dyns . . . . .	237
sum_coils . . . . .	238
sum_dyns . . . . .	238
sum_mrs . . . . .	239
sum_mrs_list . . . . .	239
svs_1h_brain_analysis . . . . .	240
svs_1h_brain_analysis_dev . . . . .	241
svs_1h_brain_batch_analysis . . . . .	243
sv_res_table . . . . .	244

td2fd . . . . .	244
tdsr . . . . .	245
td_conv_filt . . . . .	245
te . . . . .	246
tr . . . . .	246
trim_paths . . . . .	247
t_test_spec . . . . .	247
varpro_3_para_opts . . . . .	248
varpro_basic_opts . . . . .	248
varpro_opts . . . . .	249
vec2mrs_data . . . . .	250
write_basis . . . . .	251
write_basis_niidir . . . . .	251
write_basis_tqn . . . . .	252
write_mrs . . . . .	252
write_mrs_nifti . . . . .	253
write_pulse_ascii . . . . .	253
zero_fade_spec . . . . .	254
zero_higher_orders . . . . .	254
zero_td_pts_end . . . . .	255
zf . . . . .	255
zf_xy . . . . .	256

<b>Index</b>	<b>257</b>
--------------	------------

---

spant-package	<i>spant: spectroscopy analysis tools.</i>
---------------	--

---

## 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.wilson111@gmail.com> ([ORCID](#))

Authors:

- Martin Wilson <martin.wilson111@gmail.com> ([ORCID](#))

Other contributors:

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

**See Also**

Useful links:

- <https://spantdoc.wilsonlab.co.uk/>
- <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_pre = 0.078,  
  max_shift_fine = NULL,  
  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,  
  adaptive_bl_comps_pppm = FALSE,  
  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 = NULL,
max_basis_damping_broad = NULL,
ahat_calc_method = "lh_pnnls",
prefit_phase_search = TRUE,
freq_reg = NULL,
freq_reg_naa = NULL,
lb_reg = NULL,
asym_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,
lb_init_approx_fit = FALSE,
zf_offset = NULL
)

```

### 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_pre</code>	The maximum allowable global shift to be applied in the approximate (pre-fit) phases of analysis (ppm).
<code>max_shift_fine</code>	The maximum allowable global shift to be applied in the detailed fit phase of analysis (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 <code>prefit_phase_search</code> 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).
adaptive_bl_comps_pppm	adjust the spline basis density in the detailed fit phase, based on the required level of smoothness, to reduce computation time.
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	maximum 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. Larger values result in less flexible baselines.
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. The default value is set to max_basis_shift.
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. The default value is set to max_basis_damping.
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.
freq_reg_naa	frequency shift parameter for NAA and NAAG.
lb_reg	individual line broadening parameter.
asym_reg	lineshape asymmetry 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 (in Hz) for the individual basis signals. Setting to 0 will clash with the minimum allowable value (eg hard constraint) during the detailed fit.
lb_init_approx_fit	apply lb_init to the basis during the approximate iterative fit.
zf_offset	offset in number of data points from the end of the FID to zero-fill. Default is NULL and will automatically set this to 50 points when the FID distortion flag is set for the mrs_data.

**Value**

full list of options.

**Examples**

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

---

abfit\_opts\_v1\_9\_0      *Return a list of options for an ABfit analysis to maintain comparability with analyses performed with version 1.9.0 (and earlier) of spant.*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

full list of options.

---

abfit\_reg\_opts      *Return a list of options for an ABfit analysis with regularisation.*

---

**Description**

Return a list of options for an ABfit analysis with regularisation.

**Usage**

```
abfit_reg_opts(  
  init_damping = 5,  
  maxiters = 128,  
  max_shift_pre = 0.078,  
  max_shift_fine = 0.05,  
  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,  
adaptive_bl_comps_pppm = TRUE,  
export_sp_fit = FALSE,  
max_asym = Inf,  
max_basis_shift = Inf,  
max_basis_damping = Inf,  
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 = NULL,  
max_basis_damping_broad = NULL,  
ahat_calc_method = "lh_pnnls",  
prefit_phase_search = TRUE,  
freq_reg = 0.004,  
freq_reg_naa = NULL,  
lb_reg = "lcm_compat",  
asym_reg = 0.1,  
output_all_paras = FALSE,  
output_all_paras_raw = FALSE,  
input_paras_raw = NULL,  
optim_lw_only = FALSE,  
optim_lw_only_limit = 20,  
lb_init = "lcm_compat",  
lb_init_approx_fit = FALSE,  
zf_offset = NULL  
)
```

**Arguments**

init_damping	initial value of the Gaussian global damping parameter (Hz). Very poorly shimmed or high field data may benefit from a larger value.
maxiters	The maximum number of iterations to run for the detailed fit.
max_shift_pre	The maximum allowable global shift to be applied in the approximate (pre-fit) phases of analysis (ppm).
max_shift_fine	The maximum allowable global shift to be applied in the detailed fit phase of analysis (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).
adaptive_bl_comps_pppm	adjust the spline basis density in the detailed fit phase, based on the required level of smoothness, to reduce computation time.
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. Larger values result in less flexible baselines.
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. The default value is set to max_basis_shift.
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. The default value is set to max_basis_damping.
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.
freq_reg_naa	frequency shift parameter for NAA and NAAG.
lb_reg	individual line broadening parameter.
asym_reg	lineshape asymmetry 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-broadening term as a percentage of the starting value when optim_lw_only is TRUE.
lb_init	initial Lorentzian line broadening value (in Hz) for the individual basis signals. Setting to 0 will clash with the minimum allowable value (eg hard constraint) during the detailed fit.
lb_init_approx_fit	apply lb_init to the basis during the approximate iterative fit.
zf_offset	offset in number of data points from the end of the FID to zero-fill. Default is NULL and will automatically set this to 50 points when the FID distortion flag is set for the mrs_data.

**Value**

full list of options.

**Examples**

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

---

acquire

*Simulate pulse sequence acquisition.*

---

**Description**

Simulate pulse sequence acquisition.

**Usage**

```
acquire(sys, rec_phase = 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.

---

add_noise_spec_snr	<i>Add noise to an mrs_data object to match a given SNR.</i>
--------------------	--

---

**Description**

Add noise to an mrs\_data object to match a given SNR.

**Usage**

```
add_noise_spec_snr(  
  mrs_data,  
  target_snr,  
  sig_region = c(4, 0.5),  
  ref_data = NULL,  
  noise_free_input = TRUE  
)
```

**Arguments**

mrs_data	data to add noise to.
target_snr	desired spectral SNR, note this assumes the input data is noise-free, eg simulated data (unless noise_free_input is set to FALSE). Note the SNR is estimated from the first scan in the dataset and the same noise level is added to all spectra.
sig_region	spectral limits to search for the strongest spectral data point.
ref_data	measure the signal from the first scan in this reference data and apply the same target noise level to mrs_data.
noise_free_input	accounts for the reference data already containing noise when set to FALSE. Defaults to TRUE.

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

---

**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", "hanning" 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.

---

append_regs	<i>Append multiple regressor data frames into a single data frame.</i>
-------------	--

---

**Description**

Append multiple regressor data frames into a single data frame.

**Usage**

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

**Arguments**

...           input regressor data frames.

**Value**

output regressor data frame.

---

apply_axes	<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	<i>Apply a function across given dimensions of a MRS data object.</i>
-----------	---

---

**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	<i>Simulate an RF pulse on a single spin.</i>
-------------	---

---

**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,
  mrs_data = NULL,
  fs = NULL,
  ft = NULL,
  ref = NULL,
  nuc = NULL,
  fd = FALSE
)
```

**Arguments**

data_array	7d data array.
mrs_data	example data to copy acquisition parameters from.
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.

---

assign_dyns	<i>Overwrite a subset of dynamic scans.</i>
-------------	---

---

**Description**

Overwrite a subset of dynamic scans.

**Usage**

```
assign_dyns(mrs_data, subset, mrs_data_new)
```

**Arguments**

mrs_data	dynamic MRS data to be updated.
subset	vector containing indices to the dynamic scans to be assigned.
mrs_data_new	dynamic MRS data with new values.

**Value**

MRS data with updated dynamics.

---

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).

---

auto_phase_bl	<i>Perform zeroth-order phase correction based on expected baseline regions.</i>
---------------	--

---

### Description

Default arguments are appropriate for water suppressed 1H MRS of the brain. The following options are suitable for a water signal at 4.65 ppm: ppm\_start = c(5.2, 4.0), ppm\_end = c(5.3, 4.1), xlim = c(5.2, 4.1)

### Usage

```
auto_phase_bl(
  mrs_data,
  ppm_start = c(0.5, 4),
  ppm_end = c(0, 4.2),
  xlim = c(1.8, 4),
  mean_dyns = FALSE,
  ret_phase = FALSE
)
```

**Arguments**

mrs_data	an object of class mrs_data.
ppm_start	a vector of ppm values designating baseline regions.
ppm_end	a vectors of ppm values designating baseline regions.
xlim	region containing signal of interest, eg strong metabolite resonances.
mean_dyns	phase the mean spectrum and apply the same value to each dynamic.
ret_phase	return phase values (logical).

**Value**

MRS data object with corrected phase.

---

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.

---

basis2dyn_mrs_data	<i>Convert a basis object to a dynamic mrs_data object.</i>
--------------------	---

---

**Description**

Convert a basis object to a dynamic mrs\_data object.

**Usage**

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

**Arguments**

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

**Value**

a dynamic mrs\_data object.

---

basis2mrs_data	<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,
  lbs = 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.
lbs	a vector of Lorentzian line broadening terms (in Hz) 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_basis_corr_mat	<i>Estimate the correlation matrix for a basis set.</i>
---------------------	---

---

**Description**

Estimate the correlation matrix for a basis set.

**Usage**

```
calc_basis_corr_mat(basis, xlim = c(4, 0.2), zf = TRUE)
```

**Arguments**

basis	basis_set object.
xlim	spectral range to use in ppm.
zf	zero-fill the basis set.

**Value**

correlation matrix.

---

calc\_basis\_crlbs      *Estimate the CRLB for each element in a basis set.*

---

**Description**

Estimate the CRLB for each element in a basis set.

**Usage**

```
calc_basis_crlbs(  
  basis,  
  xlim = c(4, 0.2),  
  zf = TRUE,  
  sd = 1,  
  bl_comp_pppm = NULL  
)
```

**Arguments**

basis	basis_set object.
xlim	spectral range to use in ppm.
zf	zero-fill the basis set.
sd	standard deviation of the noise.
bl_comp_pppm	number spline baseline components to append per-ppm.

**Value**

a vector of predicted errors.

---

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

---

**Description**

Calculate the noise correlation between coil elements.

**Usage**

```
calc_coil_noise_cor(noise_data)
```

**Arguments**

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

**Value**

correlation matrix.

---

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

---

**Description**

Calculate the noise standard deviation for each coil element.

**Usage**

```
calc_coil_noise_sd(noise_data)
```

**Arguments**

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

**Value**

array of standard deviations.

---

calc\_design\_efficiency  
*Calculate the efficiency of a regressor data frame.*

---

**Description**

Calculate the efficiency of a regressor data frame.

**Usage**

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

**Arguments**

regressor\_df      input regressor data frame.

contrasts          a vector of contrast values.

---

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

---

**Description**

Calculate the effective dimensions of a spline smoother from lambda.

**Usage**

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

**Arguments**

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

**Value**

the effective dimension value.

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

---

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

---

**Description**

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

**Usage**

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

**Arguments**

y	array.
degree	polynomial degree.

**Value**

standard deviation of the fit residuals.

---

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

---

**Description**

Calculate the sum of squares differences between two mrs\_data objects.

**Usage**

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

**Arguments**

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

**Value**

an array of the sum of squared difference values.

---

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

---

### Description

SNR is defined as the maximum signal value divided by 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 LCModel can be run</i>
-----------	---------------------------------

---

**Description**

Check LCModel can be run

**Usage**

```
check_lcm()
```

---

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

---

**Description**

Check the TARQUIN binary can be run

**Usage**

```
check_tqn()
```

---

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_coils_mrsi_gls	<i>Combine MRSI coil data using the GLS method presented by An et al JMRI 37:1445-1450 (2013).</i>
---------------------	--

---

**Description**

Combine MRSI coil data using the GLS method presented by An et al JMRI 37:1445-1450 (2013).

**Usage**

```
comb_coils_mrsi_gls(metab, noise_pts = 30, noise_mrs = NULL)
```

**Arguments**

metab	MRSI data containing metabolite data.
noise_pts	number of points from the end of the FIDs to use for noise covariance estimation.
noise_mrs	MRS data containing noise information for each coil.

**Value**

coil combined MRSI data.

---

comb_coils_svs_gls	<i>Combine SVS coil data using the GLS method presented by An et al JMRI 37:1445-1450 (2013).</i>
--------------------	---

---

**Description**

Combine SVS coil data using the GLS method presented by An et al JMRI 37:1445-1450 (2013).

**Usage**

```
comb_coils_svs_gls(
  metab,
  ref = NULL,
  noise_pts = 256,
  noise_mrs = NULL,
  use_mean_sens = TRUE
)
```

**Arguments**

metab	MRS data containing metabolite data.
ref	MRS data containing reference data (optional).
noise_pts	number of points from the end of the FIDs to use for noise covariance estimation.
noise_mrs	MRS data containing noise information for each coil.
use_mean_sens	use the dynamic mean to estimate coil sensitivities.

**Value**

coil combined MRS data.

---

comb_fit_list_dyns	<i>Combine a list of fit results into a single fit result object across the dynamic dimension.</i>
--------------------	--

---

**Description**

Combine a list of fit results into a single fit result object across the dynamic dimension.

**Usage**

```
comb_fit_list_dyns(fit_list, dyn_basis_sets = FALSE)
```

**Arguments**

`fit_list` a list of `fit_result` objects.

`dyn_basis_sets` save all basis sets separately as a list, defaults to `FALSE`.

**Value**

fit result object.

---

comb_fit_list_fit_tables	<i>Combine all fitting data points from a list of fits into a single data frame.</i>
--------------------------	--

---

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

---

**Description**

Return the default sampling frequency in Hz.

**Usage**

```
def_fs()
```

**Value**

sampling frequency in Hz.

---

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

---

**Description**

Return the default transmitter frequency in Hz.

**Usage**

```
def_ft()
```

**Value**

transmitter frequency in Hz.

---

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

---

**Description**

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

**Usage**

```
def_N()
```

**Value**

number of data points in the spectral dimension.

---

def_nuc	<i>Return the default nucleus.</i>
---------	------------------------------------

---

**Description**

Return the default nucleus.

**Usage**

```
def_nuc()
```

**Value**

number of data points in the spectral dimension.

---

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

---

**Description**

Return the default reference value for ppm scale.

**Usage**

```
def_ref()
```

**Value**

reference value for ppm scale.

---

dicom_reader	<i>A very simple DICOM reader.</i>
--------------	------------------------------------

---

**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.

---

dyn_acq_times	<i>Return a time scale vector of acquisition times for a dynamic MRS scan. The first temporal scan is assigned a value of 0.</i>
---------------	--

---

**Description**

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

**Usage**

```
dyn_acq_times(mrs_data = NULL, tr = NULL, Ndyns = NULL, Ntrans = NULL)
```

**Arguments**

mrs_data	MRS data.
tr	repetition time.
Ndys	number of dynamic scans stored, potentially less than Ntrans if block averaging has been performed.
Ntrans	number of dynamic scans acquired.

**Value**

time scale vector in units of seconds.

---

ecc	<i>Eddy current correction.</i>
-----	---------------------------------

---

**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.

---

faceoff	<i>Deface a T1 weighted head scan using the FaceOff method described in : <a href="https://github.com/srikash/FaceOff">https://github.com/srikash/FaceOff</a>. Requires ANTs to be installed.</i>
---------	---

---

**Description**

Deface a T1 weighted head scan using the FaceOff method described in : <https://github.com/srikash/FaceOff>. Requires ANTs to be installed.

**Usage**

```
faceoff(mri_path, out_dir = NULL)
```

**Arguments**

mri_path	path to the volumetric T1 data.
out_dir	optional output directory. Defaults to the same directory as mri_path if not specified.

---

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_bids_mrs	<i>Search for MRS data files in a BIDS filesystem structure.</i>
---------------	--

---

**Description**

Search for MRS data files in a BIDS filesystem structure.

**Usage**

```
find_bids_mrs(path, output_full_path = FALSE)
```

**Arguments**

path	path to the directory containing the BIDS structure.
output_full_path	output the full normalised data paths.

**Value**

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

---

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_asy_pvoigt	<i>Fit a single asymmetric pseudo-Voigt resonance in the frequency domain.</i>
----------------	--

---

**Description**

Fit a single asymmetric pseudo-Voigt resonance in the frequency domain.

**Usage**

```
fit_asy_pvoigt(  
  mrs_data,  
  freq_ppm = 4.65,  
  xlim = c(5.2, 4.1),  
  lg_limits = c(0, 1)  
)
```

**Arguments**

mrs_data	data containing the resonance to be fit.
freq_ppm	frequency estimate (in ppm) for the resonance to be fitted.
xlim	spectral range (in ppm) where the fit will be evaluated.
lg_limits	lg lineshape parameter limits.

**Value**

list of fitting results and parameters.

---

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

*Perform a fit based analysis of MRS data.*


---

### Description

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

### Usage

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

### Arguments

<code>metab</code>	metabolite data.
<code>basis</code>	basis class object or character vector to basis file in LCModel .basis format.
<code>method</code>	'ABFIT' (default), 'VARPRO', 'VARPRO_3P', 'TARQUIN' or 'LCMODEL'.
<code>w_ref</code>	water reference data for concentration scaling (optional).
<code>opts</code>	options to pass to the analysis method.
<code>parallel</code>	perform analyses in parallel (TRUE or FALSE).
<code>cl</code>	a parallel socket cluster required to run analyses in parallel. Eg, <code>cl &lt;- parallel::makeCluster(4)</code> .
<code>time</code>	measure the time taken for the analysis to complete (TRUE or FALSE).
<code>progress</code>	option is passed to <code>plyr::alply</code> function to display a progress bar during fitting. Default value is "text", set to "none" to disable.
<code>extra</code>	an optional data frame to provide additional variables for use in subsequent analysis steps, eg id or grouping variables.

### Details

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

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

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

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

## Value

MRS analysis object.

## Examples

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

## End(Not run)
```

---

fit\_res2csv

*Write fit results table to a csv file.*

---

## Description

Write fit results table to a csv file.

## Usage

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

## Arguments

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

---

`fit_svs`*Standard SVS 1H brain analysis pipeline.*

---

**Description**

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

**Usage**

```
fit_svs(  
  input,  
  w_ref = NULL,  
  output_dir = NULL,  
  mri = NULL,  
  mri_seg = NULL,  
  deface = FALSE,  
  segment_t1 = FALSE,  
  segment_t1_method = "ants",  
  external_basis = NULL,  
  append_external_basis = FALSE,  
  p_vols = NULL,  
  format = NULL,  
  pul_seq = NULL,  
  TE = NULL,  
  TR = NULL,  
  TE1 = NULL,  
  TE2 = NULL,  
  TE3 = NULL,  
  TM = NULL,  
  append_basis = NULL,  
  remove_basis = NULL,  
  pre_align = TRUE,  
  pre_align_max_shift = 40,  
  pre_align_ref_freq = c(2.01, 3.03, 3.22),  
  pre_align_ref_amp = 1,  
  dfp_corr = TRUE,  
  dfp_corr_ref_subset = NULL,  
  output_ratio = NULL,  
  ecc = FALSE,  
  hsvd_width = NULL,  
  decimate = FALSE,  
  trunc_fid_pts = NULL,  
  fit_method = NULL,  
  fit_opts = NULL,  
  fit_subset = NULL,  
  w_ref_subset = NULL,  
  legacy_ws = FALSE,
```

```

w_att = 0.7,
w_conc = 35880,
use_basis_cache = "auto",
summary_measures = NULL,
dyn_av_block_size = NULL,
dyn_av_scheme = NULL,
dyn_av_scheme_file = NULL,
dyn_basis_lb = NULL,
dyn_basis_lg = NULL,
lcm_bin_path = NULL,
plot_ppm_xlim = NULL,
extra_output = FALSE,
verbose = FALSE,
return_fit = FALSE,
write_preproc_metab_path = NULL,
overwrite = FALSE
)

```

## Arguments

input	path or mrs_data object containing MRS data.
w_ref	path or mrs_data object containing MRS water reference data.
output_dir	directory path to output fitting results.
mri	filepath or nifti object containing anatomical MRI data.
mri_seg	filepath or nifti object containing segmented MRI data.
deface	option to apply faceoff to the mri input. Defaults to FALSE.
segment_t1	segment the t1 weighted mri file with ANTs and use the results to perform partial volume correction. Defaults to FALSE.
segment_t1_method	one of : "ants" (default), "rpyants" or "fslr".
external_basis	precompiled basis set object to use for analysis.
append_external_basis	append the external basis with the internally generated one. Useful for adding experimentally acquired baseline signals to internally simulated basis sets. Defaults to FALSE - meaning only signals from the external basis will be used in analysis.
p_vols	a numeric vector of partial volumes expressed as percentages. Defaults to 100% white matter. A voxel containing 100% gray matter tissue would use : p_vols = c(WM = 0, GM = 100, CSF = 0).
format	Override automatic data format detection. See format argument in <a href="#">read_mrs()</a> for permitted values.
pul_seq	Pulse sequence to use for basis simulation. Can be one of the following values : "press", "press_ideal", "press_shaped", "steam" or "slaser". If "press" then "press_ideal" will be assumed unless the magnetic field is stronger than 2.8 Tesla, "press_shaped" will be assumed for 2.9 Tesla and above.

TE	metabolite mrs data echo time in seconds. If not supplied this will be guessed from the metab data file.
TR	metabolite mrs data repetition time in seconds. If not supplied this will be guessed from the metab data file.
TE1	PRESS or sLASER sequence timing parameter in seconds.
TE2	PRESS or sLASER sequence timing parameter in seconds.
TE3	sLASER sequence timing parameter in seconds.
TM	STEAM mixing time parameter in seconds.
append_basis	names of extra signals to add to the default basis. Eg append_basis = c("peth", "cit"). Use get_mol_names() function to print all available signals. Cannot be used with precompiled basis sets.
remove_basis	grep expression to match names of signals to remove from the basis. For example: use "laclala" to remove lactate and alanine; "*" to remove all signals and "^mml^lip" to remove all macromolecular and lipid signals. This operation is performed before signals are added with append_basis. Cannot be used with precompiled basis sets.
pre_align	perform simple frequency alignment to known reference peaks.
pre_align_max_shift	maximum allowable shift in Hz. Defaults to 40 Hz.
pre_align_ref_freq	reference frequency in ppm units. More than one frequency may be specified. Defaults to : c(2.01, 3.03, 3.22).
pre_align_ref_amp	reference frequency relative amplitudes. Defaults to 1, corresponding to equal amplitudes.
dfp_corr	perform dynamic frequency and phase correction using the RATS method.
dfp_corr_ref_subset	specify a subset of dynamics to use as reference scans for dynamic frequency and phase correction. For example, 1:16 would use the mean of the first 16 dynamic scans as a reference spectrum.
output_ratio	optional string to specify a metabolite ratio to output. Defaults to "tCr". Multiple metabolites may be specified for multiple outputs. Set to NA to omit.
ecc	option to perform water reference based eddy current correction, defaults to FALSE.
hsvd_width	set the width of the HSVD filter in Hz. Note the applied width is between -width and +width Hz, with 0 Hz being defined at the centre of the spectral width. Default is disabled (set to NULL), 30 Hz is a reasonable value.
decimate	option on decimate the data by a factor of 2 before analysis. Defaults to FALSE.
trunc_fid_pts	number of points to truncate the input data by in the time-domain. E.g. setting to 1024 will ensure data with more time-domain points will be truncated to a length of 1024. Defaults to NULL, where truncation is not performed.
fit_method	can be "ABFIT-REG" or "LCMODEL. Defaults to "ABFIT-REG".
fit_opts	options to pass to the fitting method.

fit_subset	specify a subset of dynamics to analyse, for example 1:16 would only fit the first 16 dynamic scans.
w_ref_subset	specify a subset of dynamics to use as water reference following rats alignment and averaging. For example c(1, 2) would only use the average of the first two dynamic water reference scans. Default value of NULL uses all available water reference scans.
legacy_ws	perform and output legacy water scaling compatible with default LCModel and TARQUIN behaviour. See w_att and w_conc arguments to change the default assumptions. Default value is FALSE.
w_att	water attenuation factor (default = 0.7) for legacy water scaling. 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) for legacy water scaling. Default value corresponds to typical white matter. Set to 43300 for gray matter, and 55556 for phantom measurements.
use_basis_cache	Pre-cache basis sets to reduce analysis speed. Can be one of the following : "auto", "all" or "none". The default value of "auto" will only use the cache for 3T PRESS - which generally requires more detailed simulation due to high CSD.
summary_measures	output an additional table with a subset of metabolite levels, eg c("tNAA", "tNAA/tCr", "tNAA/tCho", "Lac/tNAA").
dyn_av_block_size	perform temporal averaging with the specified block size. Defaults to NULL, eg average across all dynamic scans.
dyn_av_scheme	a numeric vector of sequential integers (starting at 1), with the same length as the number of dynamic scans in the metabolite data. For example: c(1, 1, 2, 1, 3, 1, 1).
dyn_av_scheme_file	a file path containing a single column of sequential integers (starting at 1) with the same length as the number of dynamic scans in the metabolite data. File may be formatted as .xlsx, .xls, text or csv format.
dyn_basis_lb	dynamic basis line-broadening to apply in Hz.
dyn_basis_lg	dynamic basis Lorentz-Gauss lineshape factor between 0 and 1. Defaults to 0, pure Lorentzian.
lcm_bin_path	set the path to LCModel binary.
plot_ppm_xlim	plotting ppm axis limits in the html results. results.
extra_output	write extra output files for generating custom plots. Defaults to FALSE.
verbose	output potentially useful information.
return_fit	return a fit object, defaults to FALSE.
write_preproc_metab_path	path to write the preprocessed metabolite data in NIfTI format.
overwrite	overwrite existing fitting result files, defaults to FALSE.

## Examples

```
metab <- system.file("extdata", "philips_spar_sdat_WS.SDAT",
                    package = "spant")
w_ref <- system.file("extdata", "philips_spar_sdat_W.SDAT",
                    package = "spant")
out_dir <- file.path("~", "fit_svs_result")
## Not run:
fit_result <- fit_svs(metab, w_ref, out_dir)

## End(Not run)
```

---

fit\_svs\_edited

*Edited SVS 1H brain analysis pipeline.*

---

## Description

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

## Usage

```
fit_svs_edited(
  input,
  w_ref = NULL,
  output_dir = NULL,
  mri = NULL,
  mri_seg = NULL,
  deface = FALSE,
  segment_t1 = FALSE,
  segment_t1_method = "ants",
  external_basis = NULL,
  p_vols = NULL,
  format = NULL,
  editing_type = "gaba_1.9",
  editing_scheme = NULL,
  invert_edit_on = NULL,
  invert_edit_off = NULL,
  pul_seq = NULL,
  TE = NULL,
  TR = NULL,
  TE1 = NULL,
  TE2 = NULL,
  TE3 = NULL,
  TM = NULL,
  append_basis_ed_off = NULL,
  remove_basis_ed_off = NULL,
  pre_align = TRUE,
  dfp_corr = TRUE,
```

```

    output_ratio = NULL,
    ecc = FALSE,
    hsvd_width = NULL,
    decimate = FALSE,
    trunc_fid_pts = NULL,
    fit_opts_edited = NULL,
    fit_opts_ed_off = NULL,
    fit_subset = NULL,
    legacy_ws = FALSE,
    w_att = 0.7,
    w_conc = 35880,
    use_basis_cache = "auto",
    summary_measures = NULL,
    dyn_av_block_size = NULL,
    dyn_av_scheme = NULL,
    dyn_av_scheme_file = NULL,
    plot_ppm_xlim = NULL,
    extra_output = FALSE,
    verbose = FALSE,
    return_fit = FALSE,
    overwrite = FALSE
)

```

### Arguments

input	path or mrs_data object containing MRS data.
w_ref	path or mrs_data object containing MRS water reference data.
output_dir	directory path to output fitting results.
mri	filepath or nifti object containing anatomical MRI data.
mri_seg	filepath or nifti object containing segmented MRI data.
deface	option to apply faceoff to the mri input. Defaults to FALSE.
segment_t1	segment the t1 weighted mri file with ANTs and use the results to perform partial volume correction. Defaults to FALSE.
segment_t1_method	one of : "ants" (default), "rpyants" or "fslr".
external_basis	precompiled basis set object to use for analysis.
p_vols	a numeric vector of partial volumes expressed as percentages. Defaults to 100% white matter. A voxel containing 100% gray matter tissue would use : p_vols = c(WM = 0, GM = 100, CSF = 0).
format	Override automatic data format detection. See format argument in <a href="#">read_mrs()</a> for permitted values.
editing_type	can be one of : "gaba_1.9" or "gsh_4.54". Defaults to "gaba_1.9".
editing_scheme	describes the dynamic data ordering. Can be one of: 'on-off-blocks', 'on-off-interleaved', 'off-on-blocks' or 'off-on-interleaved'.
invert_edit_on	set to TRUE to invert the edit-on sub-spectra.

invert_edit_off	set to TRUE to invert the edit-off sub-spectra.
pul_seq	Pulse sequence to use for basis simulation. Can be one of the following values : "press", "press_ideal", "press_shaped", "steam" or "slaser". If "press" then "press_ideal" will be assumed unless the magnetic field is stronger than 2.8 Tesla, "press_shaped" will be assumed for 2.9 Tesla and above.
TE	metabolite mrs data echo time in seconds. If not supplied this will be guessed from the metab data file.
TR	metabolite mrs data repetition time in seconds. If not supplied this will be guessed from the metab data file.
TE1	PRESS or sLASER sequence timing parameter in seconds.
TE2	PRESS or sLASER sequence timing parameter in seconds.
TE3	sLASER sequence timing parameter in seconds.
TM	STEAM mixing time parameter in seconds.
append_basis_ed_off	names of extra signals to add to the default basis. Eg append_basis = c("peth", "cit"). Use get_mol_names() function to print all available signals. Cannot be used with precompiled basis sets.
remove_basis_ed_off	grep expression to match names of signals to remove from the basis. For example: use "laclala" to remove lactate and alanine; "*" to remove all signals and "^mm/^lip" to remove all macromolecular and lipid signals. This operation is performed before signals are added with append_basis_ed_off. Cannot be used with precompiled basis sets.
pre_align	perform simple frequency alignment to known reference peaks.
dfp_corr	perform dynamic frequency and phase correction using the RATS method.
output_ratio	optional string to specify a metabolite ratio to output. Defaults to "tCr". Multiple metabolites may be specified for multiple outputs. Set to NA to omit.
ecc	option to perform water reference based eddy current correction, defaults to FALSE.
hsvd_width	set the width of the HSVD filter in Hz. Note the applied width is between -width and +width Hz, with 0 Hz being defined at the centre of the spectral width. Default is disabled (set to NULL), 30 Hz is a reasonable value.
decimate	option to decimate the data by a factor of 2 before analysis. Defaults to FALSE.
trunc_fid_pts	number of points to truncate the input data by in the time-domain. E.g. setting to 1024 will ensure data with more time-domain points will be truncated to a length of 1024. Defaults to NULL, where truncation is not performed.
fit_opts_edited	options to pass to the fitting method for the edited spectrum.
fit_opts_ed_off	options to pass to the fitting method for the edit-off spectrum.
fit_subset	specify a subset of dynamics to analyse, for example 1:16 would only fit the first 16 dynamic scans.

legacy_ws	perform and output legacy water scaling compatible with default LCModel and TARQUIN behaviour. See w_att and w_conc arguments to change the default assumptions. Default value is FALSE.
w_att	water attenuation factor (default = 0.7) for legacy water scaling. 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) for legacy water scaling. Default value corresponds to typical white matter. Set to 43300 for gray matter, and 55556 for phantom measurements.
use_basis_cache	Pre-cache basis sets to reduce analysis speed. Can be one of the following : "auto", "all" or "none". The default value of "auto" will only use the cache for 3T PRESS - which generally requires more detailed simulation due to high CSD.
summary_measures	output an additional table with a subset of metabolite levels, eg c("tNAA", "tNAA/tCr", "tNAA/tCho", "Lac/tNAA").
dyn_av_block_size	perform temporal averaging with the specified block size. Defaults to NULL, eg average across all dynamic scans.
dyn_av_scheme	a numeric vector of sequential integers (starting at 1), with the same length as the number of dynamic scans in the metabolite data. For example: c(1, 1, 2, 1, 1, 3, 1, 1).
dyn_av_scheme_file	a file path containing a single column of sequential integers (starting at 1) with the same length as the number of dynamic scans in the metabolite data. File may be formatted as .xlsx, .xls, text or csv format.
plot_ppm_xlim	plotting ppm axis limits in the html results. results.
extra_output	write extra output files for generating custom plots. Defaults to FALSE.
verbose	output potentially useful information.
return_fit	return a fit object, defaults to FALSE.
overwrite	overwrite existing fitting result files, defaults to FALSE.

## Examples

```

metab <- system.file("extdata", "philips_spar_sdat_WS.SDAT",
                    package = "spant")
w_ref <- system.file("extdata", "philips_spar_sdat_W.SDAT",
                    package = "spant")
out_dir <- file.path("~", "fit_svs_result")
## Not run:
fit_result <- fit_svs(metab, w_ref, out_dir)

## End(Not run)

```

---

`fit_svs_edited_group_results`*Combine edited fitting results for group analysis.*

---

**Description**

Combine edited fitting results for group analysis.

**Usage**

```
fit_svs_edited_group_results(  
  search_path = NULL,  
  paths = NULL,  
  output_dir = "fit_svs_edited_group_results",  
  verbose = TRUE  
)
```

**Arguments**

<code>search_path</code>	path to start recursive search for fitting results. Cannot be used together with the <code>paths</code> argument.
<code>paths</code>	a set of paths to span output files, usually named : "spant_fit_svs_edited_data.rds". Cannot be used together with the <code>search_path</code> argument.
<code>output_dir</code>	directory path to store group results.
<code>verbose</code>	verbose, defaults to TRUE.

---

`fit_svs_group_results` *Combine fitting results for group analysis.*

---

**Description**

Combine fitting results for group analysis.

**Usage**

```
fit_svs_group_results(  
  search_path = NULL,  
  paths = NULL,  
  output_dir = "fit_svs_group_results",  
  verbose = TRUE  
)
```

**Arguments**

search_path	path to start recursive search for fitting results. Cannot be used together with the paths argument.
paths	a set of paths to spant output files, usually named : "spant_fit_svs_data.rds". Cannot be used together with the search_path argument.
output_dir	directory path to store group results.
verbose	verbose, defaults to TRUE.

---

fit_svs_gui	<i>GUI interface for the standard SVS 1H brain analysis pipeline, this is a work in progress, and not ready for serious use.</i>
-------------	--

---

**Description**

GUI interface for the standard SVS 1H brain analysis pipeline, this is a work in progress, and not ready for serious use.

**Usage**

```
fit_svs_gui()
```

---

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 T1 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_baseline_reg	<i>Generate baseline regressor.</i>
------------------	-------------------------------------

---

**Description**

Generate baseline regressor.

**Usage**

```
gen_baseline_reg(mrs_data = NULL, tr = NULL, Ndyns = NULL, Ntrans = NULL)
```

**Arguments**

mrs_data	mrs_data object for timing information.
tr	repetition time.
Ndyns	number of dynamic scans stored, potentially less than Ntrans if block averaging has been performed.
Ntrans	number of dynamic scans acquired.

**Value**

a single baseline regressor with value of 1.

---

gen_bold_reg	<i>Generate BOLD regressors.</i>
--------------	----------------------------------

---

**Description**

Generate BOLD regressors.

**Usage**

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

**Arguments**

onset	stimulus onset in seconds.
duration	stimulus duration in seconds.
trial_type	string label for the stimulus.
mrs_data	mrs_data object for timing information.
tr	repetition time.
Ndyns	number of dynamic scans stored, potentially less than Ntrans if block averaging has been performed.
Ntrans	number of dynamic scans acquired.
match_tr	match the output to the input mrs_data.
dt	timing resolution for internal calculations.
normalise	normalise the response function to have a maximum value of one.

**Value**

BOLD regressor data frame.

---

gen_conv_reg	<i>Generate regressors by convolving a specified response function with a stimulus.</i>
--------------	---

---

### Description

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

### Usage

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

### Arguments

onset	stimulus onset in seconds.
duration	stimulus duration in seconds.
trial_type	string label for the stimulus.
mrs_data	mrs_data object for timing information.
tr	repetition time.
Ndyns	number of dynamic scans stored, potentially less than Ntrans if block averaging has been performed.
Ntrans	number of dynamic scans acquired.
resp_fn	a data frame specifying the response function to be convolved.
match_tr	match the output to the input mrs_data.
normalise	normalise the response function to have a maximum value of one.

### Value

BOLD regressor data frame.

---

gen\_F                      *Generate the F product operator.*

---

**Description**

Generate the F product operator.

**Usage**

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

**Arguments**

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

**Value**

F product operator matrix.

---

gen\_F\_xy                      *Generate the Fxy product operator with a specified phase.*

---

**Description**

Generate the Fxy product operator with a specified phase.

**Usage**

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

**Arguments**

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

**Value**

product operator matrix.

---

gen_group_reg	<i>Expand a regressor matrix for a group analysis.</i>
---------------	--

---

**Description**

Expand a regressor matrix for a group analysis.

**Usage**

```
gen_group_reg(regressor_df, n)
```

**Arguments**

regressor_df	input regressor data frame.
n	number of datasets n the group.

---

gen_I	<i>Generate the I product operator for a single spin.</i>
-------	---

---

**Description**

Generate the I product operator for a single spin.

**Usage**

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

**Arguments**

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

**Value**

I product operator matrix.

---

gen\_impulse\_reg      *Generate impulse regressors.*

---

### Description

Generate impulse regressors.

### Usage

```
gen_impulse_reg(
  onset,
  trial_type = NULL,
  mrs_data = NULL,
  tr = NULL,
  Ndyns = NULL,
  Ntrans = NULL
)
```

### Arguments

onset	stimulus onset in seconds.
trial_type	string label for the stimulus.
mrs_data	mrs_data object for timing information.
tr	repetition time.
Ndyns	number of dynamic scans stored, potentially less than Ntrans if block averaging has been performed.
Ntrans	number of dynamic scans acquired.

### Value

impulse regressors data frame.

---

gen\_numeric\_reg      *Generate a regressor from a numeric vector.*

---

### Description

Generate a regressor from a numeric vector.

**Usage**

```
gen_numeric_reg(
  in_vec,
  name,
  mrs_data = NULL,
  tr = NULL,
  Ndyns = NULL,
  Ntrans = NULL
)
```

**Arguments**

in_vec	a numeric input vector the same length as the number of stored dynamic scans.
name	a character vector representing the regressor name.
mrs_data	mrs_data object for timing information.
tr	repetition time.
Ndyns	number of dynamic scans stored, potentially less than Ntrans if block averaging has been performed.
Ntrans	number of dynamic scans acquired.

**Value**

a single baseline regressor with value of 1.

---

gen_poly_reg	<i>Generate polynomial regressors.</i>
--------------	--

---

**Description**

Generate polynomial regressors.

**Usage**

```
gen_poly_reg(degree, mrs_data = NULL, tr = NULL, Ndyns = NULL, Ntrans = NULL)
```

**Arguments**

degree	the degree of the polynomial.
mrs_data	mrs_data object for timing information.
tr	repetition time.
Ndyns	number of dynamic scans stored, potentially less than Ntrans if block averaging has been performed.
Ntrans	number of dynamic scans acquired.

**Value**

polynomial regressors.

---

gen_trap_reg	<i>Generate trapezoidal regressors.</i>
--------------	---

---

### Description

Generate trapezoidal regressors.

### Usage

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

### Arguments

onset	stimulus onset in seconds.
duration	stimulus duration in seconds.
trial_type	string label for the stimulus.
mrs_data	mrs_data object for timing information.
tr	repetition time.
Ndyns	number of dynamic scans stored, potentially less than Ntrans if block averaging has been performed.
Ntrans	number of dynamic scans acquired.
rise_t	time to reach a plateau from baseline in seconds.
fall_t	time to fall from plateau level back to baseline in seconds.
exp_fall	model an exponential fall instead of linear.
exp_fall_power	exponential fall power.
smo_sigma	standard deviation of Gaussian smoothing kernel in seconds. Set to NULL to disable (default behavior).

match_tr	match the output to the input mrs_data.
dt	timing resolution for internal calculations.
normalise	normalise the response function to have a maximum value of one.

**Value**

trapezoidal regressor data frame.

---

get\_1h\_braino\_basis\_names

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

---

**Description**

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

**Usage**

```
get_1h_braino_basis_names()
```

**Value**

a character vector of molecule names.

---

get\_1h\_brain\_basis\_names

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

a character vector of molecule names.

---

get\_1h\_brain\_basis\_paras

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

list of mol\_parameter objects.

---

get\_1h\_brain\_basis\_paras\_v1

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

list of mol\_parameter objects.

---

get\_1h\_brain\_basis\_paras\_v2

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

list of mol\_parameter objects.

---

get\_1h\_brain\_basis\_paras\_v3

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

list of mol\_parameter objects.

get\_1h\_spectre\_basis\_names

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

**Description**

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

**Usage**

```
get_1h_spectre_basis_names()
```

**Value**

a character vector of molecule names.

get\_2d\_psf

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

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

A matrix of the PSF with 1mm resolution.

---

get_acq_paras	<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_ants_dir	<i>Return the ANTs installation directory, or throw an error if not found.</i>
--------------	--

---

**Description**

Will check and return the "spant.ants\_dir" option set by set\_ants\_dir. If not set, will search the spant\_resources directory for ANTs and return the most recent version if multiple are found.

**Usage**

```
get_ants_dir()
```

**Value**

ANTs installation directory.

---

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_hrf	<i>Generate a double gamma model of the HRF as used in SPM.</i>
---------	---

---

**Description**

Generate a double gamma model of the HRF as used in SPM.

**Usage**

```
get_hrf(end_t = 30, res_t = 0.01)
```

**Arguments**

end_t	last time point to generate in seconds.
res_t	temporal resolution in seconds, defaults to 10ms.

**Value**

a data.frame of time and HRF vectors.

---

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	<i>Generate a MRSI VOI from an mrs_data object.</i>
--------------	---

---

**Description**

Generate a MRSI VOI from an mrs\_data object.

**Usage**

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

**Arguments**

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

**Value**

volume data as a nifti object.

---

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

---

**Description**

Generate a MRSI voxel from an mrs\_data object.

**Usage**

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

**Arguments**

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

**Value**

volume data as a nifti object.

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

volume data as a nifti object.

---

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

---

**Description**

Generate an affine for nifti generation.

**Usage**

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

**Arguments**

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

**Value**

affine matrix.

---

get\_odd\_dyns            *Return odd numbered dynamic scans starting from 1 (1,3,5...).*

---

**Description**

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

**Usage**

```
get_odd_dyns(mrs_data)
```

**Arguments**

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

**Value**

dynamic MRS data containing odd numbered scans.

---

get_ref	<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_spant_resources_dir`*Return the spant resources directory.*

---

**Description**

Usually used to store additional software, datasets and templates. An attempt will be made to create the folder if not found. The default resources directory is "spant\_resources" located in the users HOME directory.

**Usage**`get_spant_resources_dir()`**Value**

full path to the spant resources directory.

---

`get_spin_num`*Return the spin number for a given nucleus.*

---

**Description**

Return the spin number for a given nucleus.

**Usage**`get_spin_num(nucleus)`**Arguments**

nucleus            nucleus name, eg "1H".

**Value**

spin number.

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

volume data as a nifti object.

---

get_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.

---

glm_spec	<i>Perform a GLM analysis of dynamic MRS data in the spectral domain.</i>
----------	---

---

**Description**

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

**Usage**

```
glm_spec(mrs_data, regressor_df, full_output = FALSE)
```

**Arguments**

mrs_data	single-voxel dynamics MRS data.
regressor_df	a data frame containing temporal regressors to be applied to each spectral data-point.
full_output	append mrs_data and regressor_df to the output list.

**Value**

list of statistical results.

---

glm\_spec\_fmrs\_fl      *Perform first-level spectral GLM analysis of an fMRS dataset.*

---

### Description

Perform first-level spectral GLM analysis of an fMRS dataset.

### Usage

```
glm_spec_fmrs_fl(  
  regressor_df,  
  analysis_dir = "spant_analysis",  
  output_dir = "spec_glm",  
  exclude_labels = NULL,  
  labels = NULL,  
  xlim = c(4, 0.2),  
  vline = c(1.35, 1.28, 2.35, 2.29),  
  lb = 4,  
  return_results = FALSE  
)
```

### Arguments

regressor_df	a data frame containing temporal regressors to be applied to each spectral data-point.
analysis_dir	directory containing preprocessed data generated by the preproc_svs_dataset function.
output_dir	directory to save glm results.
exclude_labels	vector of labels of scans to exclude, eg poor quality data.
labels	labels to describe each data set.
xlim	spectral range to include in the analysis.
vline	vertical lines to add to the plot.
lb	linebroadening to add in Hz before GLM analysis.
return_results	function will return key outputs, defaults to FALSE.

---

glm\_spec\_fmrs\_group    *Perform group-level spectral GLM analysis of an fMRS dataset.*

---

**Description**

Perform group-level spectral GLM analysis of an fMRS dataset.

**Usage**

```
glm_spec_fmrs_group(  
  regressor_df,  
  analysis_dir = "spant_analysis",  
  exclude_labels = NULL,  
  labels = NULL  
)
```

**Arguments**

regressor_df	a data frame containing temporal regressors to be applied to each spectral data-point.
analysis_dir	directory containing preprocessed data generated by the preproc_svs_dataset function.
exclude_labels	vector of labels of scans to exclude, eg poor quality data.
labels	labels to describe each data set.

---

glm\_spec\_group\_linhyp    *Test a group-level spectral GLM linear hypothesis.*

---

**Description**

Test a group-level spectral GLM linear hypothesis.

**Usage**

```
glm_spec_group_linhyp(hmat, analysis_dir = "spant_analysis")
```

**Arguments**

hmat	linear hypothesis matrix.
analysis_dir	directory containing preprocessed data generated by the preproc_svs_dataset function.

---

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,  
  hline = NULL,  
  hline_lty = 2,  
  hline_col = "white",
```

```

    vline = NULL,
    vline_lty = 2,
    vline_col = "white",
    legend = FALSE,
    ...
)

```

### 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", "time_sec", "x", "y", "z", "coil" or <code>NULL</code> . If <code>NULL</code> (the default) all spectra are collapsed into the dynamic dimension and displayed.
x_pos	the x index to plot.
y_pos	the y index to plot.
z_pos	the z index to plot.
dyn	the dynamic index to plot.
coil	the coil element number to plot.
restore_def_par	restore default plotting par values after the plot has been made.
y_ticks	a vector of indices specifying where to place additional red tick marks.
hline	add a horizontal line at the specified value.
hline_lty	linetype for the horizontal line.
hline_col	colour for the horizontal line.
vline	add a vertical line at the specified value.
vline_lty	linetype for the vertical line.
vline_col	colour for the vertical line.
legend	add a colour bar to the plot using the <code>imagePlot</code> function from the <code>fields</code> package.
...	other arguments to pass to the plot method.

---

img2kspace_xy	<i>Transform 2D MRSI data to k-space in the x-y direction.</i>
---------------	--

---

**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.

---

install_ants	<i>Install ANTs / ANTsX from : <a href="https://github.com/ANTsX/ANTs/releases">https://github.com/ANTsX/ANTs/releases</a></i>
--------------	--

---

**Description**

Install ANTs / ANTsX from : <https://github.com/ANTsX/ANTs/releases>

**Usage**

```
install_ants(platform, version = "2.6.5")
```

**Arguments**

platform	see the releases page for supported platforms. Common platforms include : "macos-14-ARM64-clang", "almalinux9-X64-gcc", "centos7-X64-gcc" or "ubuntu-24.04-X64-gcc". Note, It may be necessary to increase the timeout with "options(timeout = 1000)" on slower connections.
version	ANTs version to install, defaults to "2.6.5".

---

install_cli	<i>Install the spant command-line interface scripts to a system path.</i>
-------------	---

---

**Description**

This should be run following each new install of spant to ensure consistency. Typical command line usage : `sudo Rscript -e "spant::install_cli()"`

**Usage**

```
install_cli(path = NULL)
```

**Arguments**

path	optional path to install the scripts. Defaults to : "/usr/local/bin".
------	---

---

install_faceoff	<i>Install FaceOff from : <a href="https://github.com/srikash/FaceOff">https://github.com/srikash/FaceOff</a>, requires ANTs to be installed to work.</i>
-----------------	---

---

**Description**

Install FaceOff from : <https://github.com/srikash/FaceOff>, requires ANTs to be installed to work.

**Usage**

```
install_faceoff()
```

---

install\_oasis\_template      *Install the Oasis brain template from :*  
*<https://doi.org/10.6084/m9.figshare.915436.v2>*

---

**Description**

Install the Oasis brain template from : <https://doi.org/10.6084/m9.figshare.915436.v2>

**Usage**

install\_oasis\_template()

---

interleave\_dyns      *Interleave the first and second half of a dynamic series.*

---

**Description**

Interleave the first and second half of a dynamic series.

**Usage**

interleave\_dyns(mrs\_data)

**Arguments**

mrs\_data      dynamic MRS data.

**Value**

interleaved data.

---

int_spec	<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.

---

lb_renoise	<i>Apply line-broadening to dynamic MRS data and add normally distributed noise (renoise) to reverse any associated improvement in SNR.</i>
------------	---

---

### Description

This function mimics a strategy used by : "Bednařík P, Tkáč I, Giove F, DiNuzzo M, Deelchand DK, Emir UE, Eberly LE, Mangia S. Neurochemical and BOLD responses during neuronal activation measured in the human visual cortex at 7 Tesla. *J Cereb Blood Flow Metab.* 2015 Mar 31;35(4):601-10. doi: 10.1038/jcbfm.2014.233. PMID: 25564236; PMCID: PMC4420878." Note that `set.seed(XX)` should be used for reproducible analyses.

**Usage**

```
lb_renoise(
  mrs_data,
  lb,
  lg = NULL,
  sig_region = c(4, 0.5),
  noise_region = c(-0.5, -2.5),
  p_order = 2
)
```

**Arguments**

<code>mrs_data</code>	data to be broadened.
<code>lb</code>	amount of line-broadening in Hz, length should be equal to the number of dynamics in <code>mrs_data</code> .
<code>lg</code>	Lorentz-Gauss lineshape parameter (between 0 and 1). Defaults to a fully Lorentzian value of 0. If one value is given, will be recycled to match the number of dynamics in <code>mrs_data</code> .
<code>sig_region</code>	a ppm region to define where the maximum signal value should be estimated.
<code>noise_region</code>	a ppm region to defined where the noise level should be estimated.
<code>p_order</code>	polynomial order to fit to the noise region before estimating the standard deviation.

**Value**

line-broadened and renoised 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,
  mrs_data = NULL,
  fs = NULL,
  ft = NULL,
  ref = NULL,
  nuc = NULL,
  fd = FALSE
)
```

### Arguments

mat	data matrix.
mrs_data	example data to copy acquisition parameters from.
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.

---

match_files	<i>Match files based on a vector of input paths and a glob pattern. The glob pattern is appended to each path and should match one file only.</i>
-------------	---

---

### Description

Match files based on a vector of input paths and a glob pattern. The glob pattern is appended to each path and should match one file only.

**Usage**

```
match_files(paths, glob)
```

**Arguments**

paths           vectors of filesystem paths.  
 glob            pattern to append to each path before passing to Sys.glob.

**Value**

matched files. NA values correspond to either no match or multiple matches.

---

match_lineshape	<i>Apply Voigt line-broadening to match a reference spectrum.</i>
-----------------	---

---

**Description**

Apply Voigt line-broadening to match a reference spectrum.

**Usage**

```
match_lineshape(  
  mrs_data,  
  ref,  
  xlim,  
  init_lb = 0.2,  
  init_lg = 0.5,  
  init_amp = 1,  
  min_lb = 0,  
  max_lb = Inf,  
  min_lg = 0,  
  max_lg = 1,  
  min_amp = 0.1,  
  max_amp = 2,  
  amp_optim = TRUE  
)
```

**Arguments**

mrs\_data        data to be broadened, note the linewidth of this spectrum must be narrower than the ref spectrum.  
 ref            reference data to match.  
 xlim           spectral region to match, eg c(5.2, 4.1) could be used to match two water resonances.  
 init\_lb        initial value for the amount of line-broadening to apply (Hz).

init_lg	initial value for the Lorentz-Gauss lineshape parameter.
init_amp	initial value for the amplitude parameter.
min_lb	minimum value for the amount of line-broadening to apply (Hz).
max_lb	maximum value for the amount of line-broadening to apply (Hz).
min_lg	minimum value for the Lorentz-Gauss lineshape parameter.
max_lg	maximum value for the Lorentz-Gauss lineshape parameter.
min_amp	minimum value for the amplitude parameter.
max_amp	maximum value for the amplitude parameter.
amp_optim	flag to include amplitude adjustment in the optimisation procedure. Defaults to TRUE.

**Value**

a list containing the matched mrs\_data, difference spectra and optimisation results.

---

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, subset = NULL)
```

**Arguments**

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

**Value**

mean dynamic data.

---

mean_dyns_scheme	<i>Average sets of dynamics according to a scheme vector.</i>
------------------	---

---

**Description**

Average sets of dynamics according to a scheme vector.

**Usage**

```
mean_dyns_scheme(mrs_data, av_scheme)
```

**Arguments**

mrs_data	dynamic MRS data.
av_scheme	vector containing consecutive integer values (starting at 1) representing sets of dynamics to average. This vector must have the same length as the number of dynamic scans in mrs_data. Any elements set to NA will not contribute to averaging.

**Value**

Dynamic MRS data set containing the averaged dynamic sets.

---

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

---

**Description**

Calculate the mean of adjacent dynamic scans.

**Usage**

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

**Arguments**

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

**Value**

dynamic data averaged in blocks.

---

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

---

**Description**

Calculate the pairwise means across a dynamic data set.

**Usage**

```
mean_dyn_pairs(mrs_data)
```

**Arguments**

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

**Value**

mean dynamic data of adjacent dynamic pairs.

---

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

---

**Description**

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

**Usage**

```
mean_mrs_list(mrs_list)
```

**Arguments**

mrs\_list      list of mrs\_data objects.

**Value**

mean mrs\_data object.

---

mean\_vec\_blocks      *Calculate the mean of adjacent blocks in a vector.*

---

**Description**

Calculate the mean of adjacent blocks in a vector.

**Usage**

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

**Arguments**

x              input vector.  
block\_size      number of adjacent elements to average over.

**Value**

vector data averaged in blocks.

---

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

---

**Description**

Calculate the median dynamic data.

**Usage**

```
median_dyns(mrs_data)
```

**Arguments**

mrs\_data          dynamic MRS data.

**Value**

median dynamic data.

---

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

---

**Description**

Apply Mod operator to an MRS dataset.

**Usage**

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

**Arguments**

z                  MRS data.

**Value**

MRS data following Mod operator.

---

mod_td	<i>Apply the Modulus operator to the time-domain MRS signal.</i>
--------	--

---

**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_data2bids	<i>Create a BIDS file structure from a vector of MRS data paths or list of mrs_data objects.</i>
---------------	--

---

### Description

Create a BIDS file structure from a vector of MRS data paths or list of mrs\_data objects.

### Usage

```
mrs_data2bids(  
  mrs_data,  
  output_dir,  
  suffix = NULL,  
  sub = NULL,  
  ses = NULL,  
  task = NULL,  
  acq = NULL,  
  nuc = NULL,  
  voi = NULL,  
  rec = NULL,  
  run = NULL,  
  echo = NULL,  
  inv = NULL,  
  skip_existing = TRUE  
)
```

### Arguments

mrs_data	vector of MRS data paths or list of mrs_data objects.
output_dir	the base directory to create the BIDS structure.
suffix	optional vector of file suffixes. Default behaviour is to automatically determine these from the input data, however it is recommended that they are specified to allow more efficient skipping of existing data.
sub	optional vector of subject labels. If not specified, these will be automatically generated as a series of increasing zero-padded integer values corresponding to the mrs_data input indices.
ses	optional vector of session labels.
task	optional vector of task labels.
acq	optional vector of acquisition labels.
nuc	optional vector of nucleus labels.
voi	optional vector of volume of interest labels.
rec	optional vector of reconstruction labels.
run	optional vector of run indices.

echo	optional vector of echo time indices.
inv	optional vector of inversion indices.
skip_existing	skip any data files that have already been converted. Defaults to TRUE, set to FALSE to force an overwrite of any existing data files.

---

mrs_data2list	<i>Split MRS data containing multiple spectra into a list of single spectra datasets.</i>
---------------	---

---

### Description

Split MRS data containing multiple spectra into a list of single spectra datasets.

### Usage

```
mrs_data2list(mrs_data)
```

### Arguments

mrs_data	input MRS dataset
----------	-------------------

### Value

list of MRS datasets

---

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_data2spec_mat	<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_data2spec_mat(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.

---

mr_data2bids	<i>Create a BIDS file structure from a vector of data paths or list of mri/mrs data objects.</i>
--------------	--

---

### Description

Create a BIDS file structure from a vector of data paths or list of mri/mrs data objects.

### Usage

```
mr_data2bids(  
  mr_data,  
  suffix,  
  output_dir,  
  sub = NULL,  
  ses = NULL,  
  task = NULL,  
  acq = NULL,  
  nuc = NULL,  
  voi = NULL,  
  rec = NULL,  
  run = NULL,  
  echo = NULL,  
  inv = NULL,  
  skip_existing = TRUE,  
  mri_format = "nifti",  
  deface_mri = FALSE  
)
```

### Arguments

mr_data	vector of data paths or list of mri/mrs objects.
suffix	vector of file suffixes, eg : c("svs", "mrsi", "T1w").
output_dir	the base directory to create the BIDS structure.
sub	optional vector of subject labels. If not specified, these will be automatically generated as a series of increasing zero-padded integer values corresponding to the mrs_data input indices.
ses	optional vector of session labels.
task	optional vector of task labels.
acq	optional vector of acquisition labels.
nuc	optional vector of nucleus labels.
voi	optional vector of volume of interest labels.
rec	optional vector of reconstruction labels.
run	optional vector of run indices.

echo	optional vector of echo time indices.
inv	optional vector of inversion indices.
skip_existing	skip any data files that have already been converted. Defaults to TRUE, set to FALSE to force an overwrite of any existing data files.
mri_format	defaults to "nifti", can also be "dicom" provided the divest package is installed.
deface_mri	option to apply fsl_deface to the mri as a preprocessing step. Defaults to FALSE, requires ANTs and faceoff to be installed when TRUE.

---

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.

---

mvifftshift	<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**

```
mvifftshift(x)
```

**Arguments**

x                   matrix input.

**Value**

output matrix.

---

n2coord *Print fit coordinates from a single index.*

---

**Description**

Print fit coordinates from a single index.

**Usage**

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

**Arguments**

n	fit index.
fit_res	fit_result object.

---

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

---

**Description**

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

**Usage**

```
Ncoils(mrs_data)
```

**Arguments**

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

---

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

---

**Description**

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

**Usage**

```
Ndyns(mrs_data)
```

**Arguments**

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

---

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

---

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

---

**Description**

Return the total number of spectra in an MRS dataset.

**Usage**

Nspec(mrs\_data)

**Arguments**

mrs\_data      MRS data.

---

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

---

**Description**

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

**Usage**

Ny(mrs\_data)

**Arguments**

mrs\_data      MRS data.

---

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

---

**Description**

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

**Usage**

Nz(mrs\_data)

**Arguments**

mrs\_data      MRS data.

---

one\_page\_pdf *Export a one-page pdf of a single fit result*

---

**Description**

Export a one-page pdf of a single fit result

**Usage**

one\_page\_pdf(fit\_res, pdf\_out\_path, title = NULL)

**Arguments**

fit\_res      fit\_result object.  
pdf\_out\_path      path to the exported pdf file.  
title      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),  
  legend_axis_cex = 0.75  
)
```

**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.

mar	plot margins.
smallplot	smallplot option for positioning the colourbar.
legend_axis_cex	font expansion factor for the legend axis text.

---

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.

---

paths2df	<i>Split paths into parts based on backslash, forwardslash and dot characters and return a data frame of these parts.</i>
----------	---

---

**Description**

Split paths into parts based on backslash, forwardslash and dot characters and return a data frame of these parts.

**Usage**

```
paths2df(paths, col_num = NULL, extra_regex = NULL)
```

**Arguments**

paths	a vector of paths.
col_num	when set, only the specified column number of the data frame will be returned.
extra_regex	extra regular expression for splitting the paths. Example, "_ " could be used to additionally split on underscores and hyphen characters.

**Value**

data.frame of separated path elements.

---

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.

---

phase_ref_1h_brain	<i>Corrected zero order phase and chemical shift offset in 1H MRS data from the brain.</i>
--------------------	--

---

**Description**

Corrected zero order phase and chemical shift offset in 1H MRS data from the brain.

**Usage**

```
phase_ref_1h_brain(  
  mrs_data,  
  mean_ref = FALSE,  
  ret_corr_only = TRUE,  
  xlim = c(4, 1.9),  
  p_deg = 3,  
  sp_N = 2,  
  basis_type = "poly"  
)
```

**Arguments**

mrs_data	MRS data to be corrected.
mean_ref	apply the phase and offset of the mean spectrum to all others. Default is FALSE.
ret_corr_only	return the corrected data only.
xlim	frequency range in ppm to consider.
p_deg	polynomial baseline order.
sp_N	number of spline functions, note the true number will be sp_N + sp_deg.
basis_type	may be one of "poly" or "spline".

**Value**

corrected MRS data.

---

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,
  invert_fit = FALSE,
  ...
)
```

**Arguments**

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

---

plot.mrs\_data

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


---

**Description**

Plotting method for objects of class mrs\_data.

**Usage**

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

**Arguments**

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

---

plot_bc	<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_reg	<i>Plot regressors as an image.</i>
----------	-------------------------------------

---

**Description**

Plot regressors as an image.

**Usage**

```
plot_reg(regressor_df)
```

**Arguments**

regressor_df	input regressor data frame.
--------------	-----------------------------

---

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

---

### Description

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

### Usage

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

### Arguments

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

---

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

---

### Description

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

### Usage

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

**Arguments**

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

---

plot_slice_map	<i>Plot a slice from a 7 dimensional array.</i>
----------------	---

---

**Description**

Plot a slice from a 7 dimensional array.

**Usage**

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

**Arguments**

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

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

---

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

---

### Description

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

### Usage

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

### Arguments

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

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

---

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

---

### Description

Plot the spectral standard deviation.

### Usage

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

### Arguments

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

---

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

---

### Description

Plot a volume as an image overlay.

### Usage

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

### Arguments

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

---

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

---

### Description

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

### Usage

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

### Arguments

<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.

---

preproc_svs	<i>Preprocess and perform quality assessment of a single SVS data set.</i>
-------------	--

---

**Description**

Preprocess and perform quality assessment of a single SVS data set.

**Usage**

```
preproc_svs(
  path,
  label = NULL,
  output_dir = NULL,
  ref_inds = NULL,
  dyn_water_ref_path = NULL,
  Ntrans = NULL,
  TR = NULL
)
```

**Arguments**

path	path to the MRS data file or IMA directory.
label	a label to describe the data set.
output_dir	output directory.
ref_inds	a vector of 1-based indices for any water reference dynamic scans.
dyn_water_ref_path	path to interleaved water reference data file or IMA directory.
Ntrans	override the number of transients set in the mrs_data file.
TR	override the TR (in seconds) set in the mrs_data file.

---

preproc_svs_dataset	<i>Preprocess and perform quality assessment of one or more SVS data sets.</i>
---------------------	--

---

**Description**

Preprocess and perform quality assessment of one or more SVS data sets.

**Usage**

```
preproc_svs_dataset(
  paths,
  labels = NULL,
  output_dir = "spant_analysis",
  exclude_labels = NULL,
  overwrite = FALSE,
  ref_inds = NULL,
  dyn_water_ref_paths = NULL,
  Ntrans = NULL,
  TR = NULL,
  return_results = FALSE
)
```

**Arguments**

paths	paths to the MRS data file or IMA directory.
labels	labels to describe each data set.
output_dir	output directory.
exclude_labels	vector of labels of scans to exclude, eg poor quality data.
overwrite	overwrite saved results, defaults to FALSE.
ref_inds	a vector of 1-based indices for any water reference dynamic scans.
dyn_water_ref_paths	paths to interleaved water reference scans.

Ntrans            override the number of transients set in the mrs\_data file.  
 TR                override the TR (in seconds) set in the mrs\_data file.  
 return\_results   function will return key outputs, defaults to FALSE.

---

print.fit\_result        *Print a summary of an object of class fit\_result.*

---

### Description

Print a summary of an object of class fit\_result.

### Usage

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

### Arguments

x                    fit\_result object.  
 ...                 further arguments.

---

print.mrs\_data        *Print a summary of mrs\_data parameters.*

---

### Description

Print a summary of mrs\_data parameters.

### Usage

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

### Arguments

x                    mrs\_data object.  
 full                print all parameters (default FALSE).  
 ...                 further arguments.

---

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

---

**Description**

Get the quantum coherence matrix for a spin system.

**Usage**

```
qn_states(sys)
```

**Arguments**

sys            spin system object.

**Value**

quantum coherence number matrix.

---

rats	<i>Robust Alignment to a Target Spectrum (RATS).</i>
------	--

---

**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,  
  list_mean_ref = TRUE,  
  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

<code>mrs_data</code>	MRS data to be corrected.
<code>ref</code>	optional MRS data to use as a reference, the mean of all dynamics is used if this argument is not supplied.
<code>xlim</code>	optional frequency range to perform optimisation, set to NULL to use the full range.
<code>max_shift</code>	maximum allowable frequency shift in Hz.
<code>p_deg</code>	polynomial degree used for baseline modelling. Negative values disable baseline modelling.
<code>sp_N</code>	number of spline functions, note the true number will be <code>sp_N + sp_deg</code> .
<code>sp_deg</code>	degree of spline functions.
<code>max_t</code>	truncate the FID when longer than <code>max_t</code> to reduce time taken, set to NULL to use the entire FID.
<code>basis_type</code>	may be one of "poly" or "spline".
<code>rescale_output</code>	rescale the <code>bl_matched_spec</code> and <code>bl</code> output to improve consistency between dynamic scans.
<code>phase_corr</code>	apply phase correction (in addition to frequency). TRUE by default.
<code>ret_corr_only</code>	return the corrected <code>mrs_data</code> object only.
<code>zero_freq_shift_t0</code>	perform a linear fit to the frequency shifts and set the (linearly modeled) shift to be 0 Hz for the first dynamic scan.
<code>list_mean_ref</code>	if <code>ref</code> is not specified and a list is provided, use the list mean scan as reference. Otherwise use the mean of each list element as its own reference.
<code>remove_freq_outliers</code>	remove dynamics based on their frequency shift.
<code>freq_outlier_thresh</code>	threshold to remove frequency outliers.
<code>remove_phase_outliers</code>	remove dynamics based on their phase shift.
<code>phase_outlier_thresh</code>	threshold to remove phase outliers.
<code>remove_amp_outliers</code>	remove dynamics based on their amplitude change.
<code>amp_outlier_thresh</code>	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	<i>Apply Re operator to an MRS dataset.</i>
-------------	---

---

**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_basis_niidir	<i>Read a basis folder containing one NIFTI MRS file for each basis element.</i>
-------------------	--

---

**Description**

Read a basis folder containing one NIFTI MRS file for each basis element.

**Usage**

```
read_basis_niidir(basis_dir)
```

**Arguments**

basis_dir	directory of basis files.
-----------	---------------------------

**Value**

a basis\_set object.

---

read_dkd_moco_log	<i>Read the log from Dinesh's MoCo sLASER sequence.</i>
-------------------	---

---

**Description**

Read the log from Dinesh's MoCo sLASER sequence.

**Usage**

```
read_dkd_moco_log(path, date, end_time)
```

**Arguments**

path	path to the log file.
date	scan date, eg "01-01-2025".
end_time	scan time, eg "14:00".

**Value**

data.frame of moco parameters.

---

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 the filesystem.</i>
----------	---

---

**Description**

Read MRS data from the filesystem.

**Usage**

```

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

```

**Arguments**

path	file name or directory containing the 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, or will be assumed to be a Siemens ima dynamic data if the path is a directory.
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.
fid_filt_dist	indicate if the data has a distorted FID due to a brick-wall filter being used to downsample the data. Default is to auto detect this from the data, but TRUE or FALSE options can be given to override detection.

**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**

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

**Value**

MRS data object.

**Examples**

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

## End(Not run)
```

---

read\_pulse\_ascii      *Read an ASCII formatted pulse file.*

---

**Description**

Read an ASCII formatted pulse file.

**Usage**

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

**Arguments**

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

**Value**

pulse waveform and header.

---

read\_pulse Bruker      *Read a Bruker formatted pulse file*

---

**Description**

Read a Bruker formatted pulse file

**Usage**

```
read_pulse Bruker(fname)
```

**Arguments**

fname                  Bruker formatted pulse file path.

**Value**

pulse waveform and header.

---

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

---

**Description**

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

**Usage**

```
read_pulse_pta(fname)
```

**Arguments**

fname	pta formatted pulse file path.
-------	--------------------------------

**Value**

pulse waveform and header.

---

read_siemens_txt_hdr	<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 = FALSE, 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_legacy	<i>Apply water reference scaling to a fitting results object to yield metabolite quantities in units of "mmol per Kg wet weight".</i>
------------------	---

---

**Description**

See the LCModel manual (section 10.2) on water-scaling for details on the assumptions and relevant references. Use this type of concentration scaling to compare fit results with LCModel and TARQUIN defaults. Otherwise scale\_amp\_molal\_pvc is the preferred method. Note, the LCModel manual (section 1.3) states:

**Usage**

```
scale_amp_legacy(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.

**Details**

"Concentrations should be labelled 'mmol per Kg wet weight'. We use the shorter (incorrect) abbreviation mM. The actual mM is the mmol per Kg wet weight multiplied by the specific gravity of the tissue, typically 1.04 in brain."

**Value**

a fit\_result object with a rescaled results table.

---

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

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

### Value

A `fit_result` object with a rescaled results table.

---

scale_amp_molal_pvc	<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

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

### Value

A fit\_result object with a rescaled results table.

---

scale_amp_molar	<i>Apply water reference scaling to a fitting results object to yield metabolite quantities in millimolar (mM) units (mol / Litre of tissue). This function is deprecated, please use scale_amp_legacy instead.</i>
-----------------	---

---

### Description

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

**Usage**

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

**Arguments**

fit_result	a result object generated from fitting.
ref_data	water reference MRS data object.
w_att	water attenuation factor (default = 0.7). Assumes water T2 of 80ms and a TE = 30 ms. $\exp(-30\text{ms} / 80\text{ms}) \sim 0.7$ .
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

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

a fit\_result object with a rescaled results table.

---

scale_amp_ratio	<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	<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_value(fit_result, value)
```

**Arguments**

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

**Value**

a `fit_result` object with a rescaled results table.

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

a fit\_result object with a rescaled results table.

---

scale\_basis\_amp      *Scale a basis object by a scalar.*

---

**Description**

Scale a basis object by a scalar.

**Usage**

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

**Arguments**

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

**Value**

basis\_set object multiplied by the amplitude scale factor.

---

scale\_basis\_from\_singlet

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

---

### Description

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

### Usage

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

### Arguments

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

### Value

a scaled basis.

---

scale\_mrs\_amp

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

---

### Description

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

### Usage

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

### Arguments

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

### Value

mrs\_data object multiplied by the amplitude scale factor.

---

scale_spec	<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 = NULL,
  ret_scale_factor = FALSE
)
```

**Arguments**

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

**Value**

normalised data.

---

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

---

**Description**

Calculate the standard deviation spectrum from an mrs\_data object.

**Usage**

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

**Arguments**

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

**Value**

sd `mrs_data` object.

---

sd.mrs\_data

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

sd `mrs_data` object.

---

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

---

**Description**

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

**Usage**

```
seconds(mrs_data)
```

**Arguments**

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

**Value**

time scale vector in units of seconds.

---

segment_t1_ants	<i>Segment T1 weighted MRI data using the ANTs binaries and write to file.</i>
-----------------	--

---

**Description**

Segment T1 weighted MRI data using the ANTs binaries and write to file.

**Usage**

```
segment_t1_ants(mri_path, out_dir = NULL)
```

**Arguments**

mri_path	path to the volumetric T1 data.
out_dir	optional output directory. Defaults to the same directory as mri_path if not specified.

---

segment_t1_fsl	<i>Segment T1 weighted MRI data using FSL FAST and write to file. Runs bet as a preprocessing step by default.</i>
----------------	--

---

### Description

This function requires a working installation of FSL and uses the fsLr package. You may need to specify the fsl install directory, eg: 'options(fsl.path = "/path/to/fsl")'

### Usage

```
segment_t1_fsl(mri_path, out_dir = NULL, deface = FALSE, bet_fit = 0.5)
```

### Arguments

mri_path	path to the volumetric T1 data.
out_dir	optional output directory. Defaults to the same directory as mri_path if not specified.
deface	deface the input T1 data before analysis. Defaults to FALSE.
bet_fit	fractional intensity threshold for bet brain extraction. Values should be between 0 and 1. Defaults to 0.5 with smaller values giving larger brain estimates.

---

segment_t1_rpyants	<i>Segment T1 weighted MRI data using the rpyANTs interface to ANTs and write to file.</i>
--------------------	--

---

### Description

Segment T1 weighted MRI data using the rpyANTs interface to ANTs and write to file.

### Usage

```
segment_t1_rpyants(mri_path, out_dir = NULL)
```

### Arguments

mri_path	path to the volumetric T1 data.
out_dir	optional output directory. Defaults to the same directory as mri_path if not specified.

---

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\_2d\_shaped     *PRESS sequence with shaped refocusing pulses.*

---

**Description**

PRESS sequence with shaped refocusing pulses.

**Usage**

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

**Arguments**

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

TE2	TE2 sequence parameter in seconds.
pulse_file	path to refocusing pulse file.
pulse_dur	refocusing pulse duration.
pulse_file_format	file format for the refocusing pulse.
refoc_flip_angle	refocusing pulse flip angle in degrees (defaults to 180).
xy_pulse_ppm	a vector of ppm values for the offset of each sub-simulation.
resamp	option to resample the pulse shape.
fs_resamp	sampling frequency (Hz) to resample.

**Value**

list of resonance amplitudes and frequencies.

---

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

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

---

**Description**

sLASER sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

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

---

**Description**

Spin echo sequence with ideal pulses.

**Usage**

```
seq_spin_echo_ideal(spin_params, ft, ref, nuc = "1H", TE = 0.03)
```

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

seq\_steam\_ideal     *STEAM sequence with ideal pulses.*

---

**Description**

STEAM sequence with ideal pulses.

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

list of resonance amplitudes and frequencies.

---

set_ants_dir	<i>Set the ANTs installation directory location.</i>
--------------	--

---

**Description**

Set the ANTs installation directory location.

**Usage**

```
set_ants_dir(dir)
```

**Arguments**

dir	ANTs installation directory.
-----	------------------------------

---

set\_def\_acq\_paras      *Set the default acquisition parameters.*

---

**Description**

Set the default acquisition parameters.

**Usage**

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

**Arguments**

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

---

set\_lcm\_cmd      *Set the command to run the LCModel command-line program.*

---

**Description**

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

**Usage**

```
set_lcm_cmd(cmd)
```

**Arguments**

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

---

set_lw	<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_asy_pvoigt	<i>Generate an asymmetric pseudo-Voigt resonance in the frequency domain.</i>
----------------	---

---

### Description

Method is described in detail by Stancik AL and Brauns EB: "A simple asymmetric lineshape for fitting infrared absorption spectra." *Vib Spectrosc.* 2008; 47: 66-69.

### Usage

```
sim_asy_pvoigt(
  freq = 0,
  fwhm = 0,
  lg = 0,
  asy = 0,
  acq_paras = def_acq_paras(),
  gen_im_pts = FALSE
)
```

### Arguments

freq	resonance frequency in ppm.
fwhm	resonance FWHM in Hz.
lg	Lorentz-Gauss lineshape parameter (between 0 and 1).
asy	asymmetry parameter.
acq_paras	list of acquisition parameters. See <a href="#">def_acq_paras</a>
gen_im_pts	option to generate imaginary data points, defaults to FALSE.

---

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,
  auto_scale = FALSE,
```

```

    use_basis_cache = FALSE,
    verbose = FALSE,
    ...
)

```

### Arguments

mol_list	list of mol_parameter objects. Alternatively, a character vector matching molecules may also be provided. Use the get_mol_names function for a full list of molecules.
pul_seq	pulse sequence function to use.
acq_paras	list of acquisition parameters or an mrs_data object. See <a href="#">def_acq_paras</a>
xlim	ppm range limiting signals to be simulated.
auto_scale	scale the basis based on the intensity of a singlet resonance. Needed for sequences with spatial simulation.
use_basis_cache	create and use a cache of simulated basis sets stored in the "basis_cache" folder in the spant resources directory. Defaults to FALSE.
verbose	output simulation progress and timings.
...	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_paras = def_acq_paras(),
  xlim = c(0.5, 4.2),
  lcm_compat = FALSE,
  TE1 = 0.01,
  TE2 = 0.02
)
```

**Arguments**

acq_paras	list of acquisition parameters or an mrs_data object. See <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.
TE1	TE1 of PRESS sequence (TE = TE1 + TE2).
TE2	TE2 of PRESS sequence.

**Value**

basis object.

---

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

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

basis object.

---

sim\_basis\_tqn            *Simulate a basis file using TARQUIN.*

---

**Description**

Simulate a basis file using TARQUIN.

**Usage**

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

**Arguments**

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

**Examples**

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

## End(Not run)
```

---

sim_brain_1h	<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 mrs_data object. See <a href="#">def_acq_paras</a> .
type	type of spectrum, only "normal" is implemented currently.
pul_seq	pulse sequence function to use.
xlim	range of frequencies to simulate in ppm.
full_output	when FALSE (default) only output the simulated MRS data. When TRUE output a list containing the MRS data, basis set object and corresponding amplitudes.
amps	a vector of basis amplitudes may be specified to modify the output spectrum.
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(),  
  nuc = def_nuc(),  
  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.
nuc	resonant nucleus.
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

<code>bw</code>	top-hat bandwidth (Hz).
<code>sigma</code>	Gaussian width smoothing parameter (Hz).
<code>fa</code>	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(),
  nuc = def_nuc(),
  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.
nuc	resonant nucleus.
dyns	number of dynamic scans to generate.

**Value**

mrs\_data object.

---

smooth\_dyns

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

---

**Description**

Smooth data across the dynamic dimension with a Gaussian kernel.

**Usage**

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

**Arguments**

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

**Value**

smoothed mrs\_data object.

sort\_basis                    *Sort the basis-set elements alphabetically.*

---

**Description**

Sort the basis-set elements alphabetically.

**Usage**

```
sort_basis(basis)
```

**Arguments**

basis                    input basis.

**Value**

sorted basis.

---

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

---

**Description**

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

**Usage**

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

**Arguments**

noise\_reps                number of spectra to fit with differing noise samples.

return\_res                return a list of fit\_result objects.

opts                      ABfit options structure.

---

spant\_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.

---

spant\_sim\_fmrs\_dataset

*Simulate an example fMRS dataset for a block design fMRS experiment and export a BIDS structure.*

---

### Description

Simulate an example fMRS dataset for a block design fMRS experiment and export a BIDS structure.

### Usage

```
spant_sim_fmrs_dataset(output_dir = NULL)
```

### Arguments

output_dir	output directory for the BIDS data. Defaults to : "HOME/sim_fmrs_dataset/data".
------------	---

---

spec_decomp	<i>Decompose an mrs_data object into white and gray matter spectra.</i>
-------------	---

---

**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", "max_cplx", "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, precomp_jc_H = NULL, precomp_Iz = NULL)
```

**Arguments**

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

**Value**

spin system object.

---

spm_pve2categorical	<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,
    invert_fit = FALSE,
    ...
)

```

### Arguments

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



xlim	the range of values to display on the x-axis, eg xlim = c(4,1).
mode	representation of the complex numbers to be plotted, can be one of: "re", "im", "mod" or "arg".
x_units	the units to use for the x-axis, can be one of: "ppm", "hz", "points" or "seconds".
fd	display data in the frequency-domain (default), or time-domain (logical).
col	set the colour of the line, eg col = rgb(1, 0, 0, 0.5).
alpha	set the line transparency, eg alpha = 0.5 is 50% transparency. Overrides any transparency levels set by col.
x_offset	separate plots in the x-axis direction by this value. Default value is 0.
y_offset	separate plots in the y-axis direction by this value.
plot_dim	the dimension to display on the y-axis, can be one of: "dyn", "x", "y", "z", "coil" or NULL. If NULL (the default) all spectra will be collapsed into the dynamic dimension and displayed.
x_pos	the x index to plot.
y_pos	the y index to plot.
z_pos	the z index to plot.
dyn	the dynamic index to plot.
coil	the coil element number to plot.
bty	option to draw a box around the plot. See ?par.
labels	add labels to each data item.
lab_cex	label size.
bl_lty	linetype for the y = 0 baseline trace. A default value NULL results in no baseline being plotted.
bl_lwd	linewidth for the y = 0 baseline trace. Defaults to 0.5.
restore_def_par	restore default plotting par values after the plot has been made.
show_grid	plot gridlines behind the data (logical). Defaults to TRUE.
grid_nx	number of cells of the grid in x and y direction. When NULL the grid aligns with the tick marks on the corresponding default axis (i.e., tickmarks as computed by axTicks). When NA, no grid lines are drawn in the corresponding direction.
grid_ny	as above.
lwd	plot linewidth.
vline	x-value to draw a vertical line.
vline_lty	linetype for the vertical line.
vline_col	colour for the vertical line.
mar	option to adjust the plot margins. See ?par.
y_scale	option to display the y-axis values (logical).
...	other arguments to pass to the matplot method.

---

subtract_rest_task	<i>Subtract mean rest spectrum from mean task spectrum after applying optimal linebroadening to the mean task spectrum. Usually used to correct for the BOLD lineshape narrowing effect in 1H fMRS data.</i>
--------------------	--

---

**Description**

Subtract mean rest spectrum from mean task spectrum after applying optimal linebroadening to the mean task spectrum. Usually used to correct for the BOLD lineshape narrowing effect in 1H fMRS data.

**Usage**

```
subtract_rest_task(mrs_data, task_vec, xlim = c(2.11, 1.91))
```

**Arguments**

mrs_data	dynamic MRS data.
task_vec	a logical vector with the same length and dynamic scans. Elements set to TRUE and FALSE will be assigned to task and rest respectively.
xlim	spectral region to match, eg c(2.11, 1.91) for tNAA region.

**Value**

a list containing the matched mrs\_data, difference spectra and optimisation results.

---

sub_even_from_odd_dyns	<i>Subtract the even dynamic scans from the odd dynamic scans.</i>
------------------------	--

---

**Description**

Subtract the even dynamic scans from the odd dynamic scans.

**Usage**

```
sub_even_from_odd_dyns(mrs_data)
```

**Arguments**

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

**Value**

subtracted result.

---

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.

---

sub_median_dyns	<i>Subtract the median dynamic spectrum from a dynamic series.</i>
-----------------	--

---

**Description**

Subtract the median dynamic spectrum from a dynamic series.

**Usage**

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

**Arguments**

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

**Value**

subtracted data.

---

sub_odd_from_even_dyns	<i>Subtract the odd dynamic scans from the even dynamic scans.</i>
------------------------	--

---

**Description**

Subtract the odd dynamic scans from the even dynamic scans.

**Usage**

```
sub_odd_from_even_dyns(mrs_data)
```

**Arguments**

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

**Value**

subtracted result.

---

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\_analysis\_dev

*Standard SVS 1H brain analysis pipeline.*

---

### Description

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

### Usage

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

**Arguments**

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

**Examples**

```

metab <- system.file("extdata", "philips_spar_sdat_WS.SDAT",
                    package = "spant")
w_ref <- system.file("extdata", "philips_spar_sdat_W.SDAT",
                    package = "spant")

## Not run:
fit_result <- svs_1h_brain_analysis(metab, w_ref, "fit_res_dir")

## End(Not run)

```

---

`svs_1h_brain_batch_analysis`*Batch interface to the standard SVS 1H brain analysis pipeline.*

---

## Description

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

## Usage

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

## Arguments

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

## Value

a list of `fit_result` objects.

---

sv_res_table	<i>Output a table of fit amplitudes and error estimates for a single-voxel fit.</i>
--------------	---

---

**Description**

Output a table of fit amplitudes and error estimates for a single-voxel fit.

**Usage**

```
sv_res_table(fit_res, format_out = FALSE)
```

**Arguments**

fit_res	input vector.
format_out	reduce the accuracy of values to aid table formatting.

**Value**

data.frame of values.

---

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.

---

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

---

**Description**

Return the echo time of an MRS dataset.

**Usage**

te(mrs\_data)

**Arguments**

mrs\_data      MRS data.

**Value**

echo time in seconds.

---

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

---

**Description**

Return the repetition time of an MRS dataset.

**Usage**

tr(mrs\_data)

**Arguments**

mrs\_data      MRS data.

**Value**

repetition time in seconds.

---

trim_paths	<i>Trim a vector of filesystem paths.</i>
------------	---

---

**Description**

Trim a vector of filesystem paths.

**Usage**

```
trim_paths(paths, dir = 0, char = 0)
```

**Arguments**

paths	vectors of filesystem paths.
dir	number of times to apply the base dirname function.
char	number of characters to trim from the end of each path. Note this is performed after the dir based trimming.

**Value**

a vector of trimmed paths.

---

t_test_spec	<i>Perform a t-test on spectral data points.</i>
-------------	--

---

**Description**

Perform a t-test on spectral data points.

**Usage**

```
t_test_spec(mrs_data, group)
```

**Arguments**

mrs_data	an mrs_data object with spectra in the dynamic dimension.
group	vector describing the group membership of each dynamic spectrum.

**Value**

a list of statistical results.

---

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

---

### Description

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

### Usage

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

### Arguments

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

### Value

list of options.

---

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

---

### Description

Return a list of options for a basic VARPRO analysis.

### Usage

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

**Arguments**

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

**Value**

full list of options.

---

varpro_opts	<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,  
  mrs_data = NULL,  
  fs = NULL,  
  ft = NULL,  
  ref = NULL,  
  nuc = NULL,  
  dyns = 1,  
  fd = FALSE  
)
```

**Arguments**

vec	the data vector.
mrs_data	example data to copy acquisition parameters from.
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 vector 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_niidir	<i>Write a basis object to folder containing one NIfTI MRS file for each basis element.</i>
--------------------	---

---

**Description**

Write a basis object to folder containing one NIfTI MRS file for each basis element.

**Usage**

```
write_basis_niidir(basis, basis_dir)
```

**Arguments**

basis	basis object to be exported.
basis_dir	directory for basis files to be written.

---

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.

---

write_pulse_ascii	<i>Write an ASCII formatted pulse file.</i>
-------------------	---

---

**Description**

Write an ASCII formatted pulse file.

**Usage**

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

**Arguments**

pulse	pulse data object.
path	file path for export.

---

zero_fade_spec	<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

abfit\_opts, 11, 15  
abfit\_opts\_v1\_9\_0, 15  
abfit\_reg\_opts, 15  
acquire, 19  
add\_noise, 20  
add\_noise\_spec\_snr, 20  
align, 21  
apodise\_xy, 22  
append\_basis, 22  
append\_coils, 23  
append\_dyns, 23  
append\_regs, 24  
apply\_axes, 24  
apply\_mrs, 25  
apply\_pulse, 25  
Arg.mrs\_data, 26  
array2mrs\_data, 26  
assign\_dyns, 27  
auto\_phase, 28  
auto\_phase\_bl, 28  
  
back\_extrap\_ar, 29  
basis2dyn\_mrs\_data, 30  
basis2mrs\_data, 30  
bbase, 31  
bc\_als, 32  
bc\_constant, 32  
bc\_gauss, 33  
bc\_poly, 33  
bc\_spline, 34  
beta2lw, 34  
bin\_spec, 35  
  
calc\_basis\_corr\_mat, 35  
calc\_basis\_crlbs, 36  
calc\_coil\_noise\_cor, 36  
calc\_coil\_noise\_sd, 37  
calc\_design\_efficiency, 37  
calc\_ed\_from\_lambda, 38  
calc\_peak\_info\_vec, 38  
  
calc\_sd\_poly, 39  
calc\_spec\_diff, 39  
calc\_spec\_snr, 40  
check\_lcm, 41  
check\_tqn, 41  
circ\_mask, 41  
coherence\_filter, 42  
collapse\_to\_dyns, 42  
comb\_coils, 43  
comb\_coils\_mrsi\_gls, 44  
comb\_coils\_svs\_gls, 44  
comb\_fit\_list\_dyns, 45  
comb\_fit\_list\_fit\_tables, 45  
comb\_fit\_list\_result\_tables, 46  
comb\_fit\_tables, 47  
comb\_metab\_ref, 47  
Conj.mrs\_data, 48  
conv\_mrs, 48  
crop\_basis, 49  
crop\_spec, 49  
crop\_td\_pts, 50  
crop\_td\_pts\_end, 50  
crop\_td\_pts\_pot, 51  
crop\_xy, 51  
crossprod\_3d, 52  
  
decimate\_mrs\_fd, 52  
decimate\_mrs\_td, 53  
deconv\_mrs, 53  
def\_acq\_paras, 54, 191, 216–220, 223  
def\_fs, 55  
def\_ft, 55  
def\_N, 55  
def\_nuc, 56  
def\_ref, 56  
dicom\_reader, 56  
diff\_mrs, 57  
downsample\_mrs\_fd, 58  
downsample\_mrs\_td, 58  
dyn\_acq\_times, 59

`ecc`, 59  
`elliptical_mask`, 60  
`est_noise_sd`, 61

`faceoff`, 61  
`fd2td`, 62  
`fd_conv_filt`, 62  
`fd_gauss_smo`, 63  
`find_bids_mrs`, 63  
`find_mrs_files`, 64  
`fit_amps`, 64  
`fit_asy_pvoigt`, 65  
`fit_diags`, 65  
`fit_mrs`, 66  
`fit_res2csv`, 67  
`fit_svs`, 68  
`fit_svs_edited`, 72  
`fit_svs_edited_group_results`, 76  
`fit_svs_group_results`, 76  
`fit_svs_gui`, 77  
`fit_t1_ti_array`, 77  
`fit_t1_tr_array`, 78  
`fit_t2_te_array`, 79  
`fp_phase`, 79  
`fp_phase_correct`, 80  
`fp_scale`, 80  
`fs`, 81  
`ft_dyns`, 81  
`ft_shift`, 82  
`ft_shift_mat`, 82

`gausswin_2d`, 83  
`gen_baseline_reg`, 83  
`gen_bold_reg`, 84  
`gen_conv_reg`, 85  
`gen_F`, 86  
`gen_F_xy`, 86  
`gen_group_reg`, 87  
`gen_I`, 87  
`gen_impulse_reg`, 88  
`gen_numeric_reg`, 88  
`gen_poly_reg`, 89  
`gen_trap_reg`, 90  
`get_1h_brain_basis_names`, 91  
`get_1h_brain_basis_paras`, 92  
`get_1h_brain_basis_paras_v1`, 92  
`get_1h_brain_basis_paras_v2`, 93  
`get_1h_brain_basis_paras_v3`, 93  
`get_1h_braino_basis_names`, 91  
`get_1h_spectre_basis_names`, 94  
`get_2d_psf`, 94  
`get_acq_paras`, 95  
`get_ants_dir`, 95  
`get_basis_subset`, 96  
`get_dyns`, 96  
`get_even_dyns`, 97  
`get_fh_dyns`, 97  
`get_fit_map`, 98  
`get_fp`, 98  
`get_gaussian_pulse`, 99  
`get_head_dyns`, 99  
`get_hrf`, 100  
`get_lcm_cmd`, 100  
`get_metab`, 100  
`get_mol_names`, 101  
`get_mol_paras`, 101  
`get_mrs_affine`, 104  
`get_mrsi2d_seg`, 102  
`get_mrsi_voi`, 102  
`get_mrsi_voxel`, 103  
`get_mrsi_voxel_xy_psf`, 103  
`get_odd_dyns`, 104  
`get_ref`, 105  
`get_seg_ind`, 105  
`get_sh_dyns`, 106  
`get_slice`, 106  
`get_spant_resources_dir`, 107  
`get_spin_num`, 107  
`get_subset`, 108  
`get_svs_voi`, 109  
`get_tail_dyns`, 109  
`get_td_amp`, 110  
`get_tqn_cmd`, 110  
`get_uncoupled_mol`, 111  
`get_voi_cog`, 111  
`get_voi_seg`, 112  
`get_voi_seg_psf`, 112  
`get_voxel`, 113  
`glm_spec`, 113  
`glm_spec_fmrs_f1`, 114  
`glm_spec_fmrs_group`, 115  
`glm_spec_group_linhyp`, 115  
`grid_shift_xy`, 117  
`gridplot`, 116  
`gridplot.mrs_data`, 116

`hsvd`, 117  
`hsvd_filt`, 118

- hsvd\_vec, 119
- hz, 119
  
- ift\_shift, 120
- ift\_shift\_mat, 120
- Im.mrs\_data, 121
- image.mrs\_data, 121
- img2kspace\_xy, 123
- Imzap, 123
- install\_ants, 124
- install\_cli, 124
- install\_faceoff, 124
- install\_oasis\_template, 125
- int\_spec, 126
- interleave\_dyns, 125
- inv\_even\_dyns, 126
- inv\_odd\_dyns, 127
- is.def, 127
- is\_fd, 128
  
- kspace2img\_xy, 128
  
- l2\_reg, 129
- lb, 130
- lb\_renoise, 130
- lofdc, 131
- lw2alpha, 132
- lw2beta, 133
  
- make\_basis\_from\_raw, 133
- mask\_dyns, 134
- mask\_fit\_res, 134
- mask\_xy, 135
- mask\_xy\_corners, 135
- mask\_xy\_ellipse, 136
- mask\_xy\_mat, 136
- mat2mrs\_data, 137
- match\_files, 137
- match\_lineshape, 138
- matexp, 139
- max\_mrs, 140
- max\_mrs\_interp, 140
- mean.list, 141
- mean.mrs\_data, 141
- mean\_dyn\_blocks, 143
- mean\_dyn\_pairs, 143
- mean\_dyns, 142
- mean\_dyns\_scheme, 142
- mean\_mrs\_list, 144
  
- mean\_vec\_blocks, 144
- median\_dyns, 145
- Mod.mrs\_data, 145
- mod\_td, 146
- mr\_data2bids, 150
- mrs\_data2basis, 146
- mrs\_data2bids, 147
- mrs\_data2list, 148
- mrs\_data2mat, 148
- mrs\_data2spec\_mat, 149
- mrs\_data2vec, 149
- mvfftshift, 151
- mvifftshift, 151
  
- n2coord, 152
- Ncoils, 152
- Ndyns, 152
- nifti\_flip\_lr, 153
- Npts, 153
- Nspec, 154
- Ntrans, 154
- Nx, 154
- Ny, 155
- Nz, 155
  
- one\_page\_pdf, 155
- ortho3, 156
- ortho3\_inter, 157
  
- paths2df, 158
- peak\_info, 158
- pg\_extrap\_xy, 159
- phase, 160
- phase\_ref\_1h\_brain, 160
- plot.fit\_result, 161
- plot.mrs\_data, 162
- plot\_bc, 165
- plot\_reg, 165
- plot\_slice\_fit, 166
- plot\_slice\_fit\_inter, 166
- plot\_slice\_map, 167
- plot\_slice\_map\_inter, 168
- plot\_spec\_sd, 169
- plot\_voi\_overlay, 169
- plot\_voi\_overlay\_seg, 170
- ppm, 170
- precomp, 171
- preproc\_svs, 171
- preproc\_svs\_dataset, 172

print.fit\_result, 173  
 print.mrs\_data, 173  
  
 qn\_states, 174  
  
 rats, 174  
 Re.mrs\_data, 176  
 re\_weighting, 191  
 read\_basis, 176  
 read\_basis\_niidir, 177  
 read\_dkd\_moco\_log, 177  
 read\_ima\_coil\_dir, 178  
 read\_ima\_dyn\_dir, 178  
 read\_lcm\_coord, 179  
 read\_mrs, 179  
 read\_mrs(), 69, 73  
 read\_mrs\_tqn, 181  
 read\_pulse\_ascii, 182  
 read\_pulse\_bruker, 182  
 read\_pulse\_pta, 183  
 read\_siemens\_txt\_hdr, 183  
 read\_tqn\_fit, 184  
 read\_tqn\_result, 184  
 recon\_imag, 185  
 recon\_imag\_vec, 185  
 recon\_twix\_2d\_mrsi, 186  
 rectangular\_mask, 186  
 rep\_array\_dim, 187  
 rep\_dyn, 187  
 rep\_mrs, 188  
 resample\_basis, 188  
 resample\_img, 189  
 resample\_voi, 189  
 reslice\_to\_mrs, 190  
 reson\_table2mrs\_data, 190  
 rm\_dyns, 191  
  
 scale\_amp\_legacy, 192  
 scale\_amp\_molal, 193  
 scale\_amp\_molal\_pvc, 194  
 scale\_amp\_molar, 194  
 scale\_amp\_molar2molal\_pvc, 195  
 scale\_amp\_ratio, 196  
 scale\_amp\_ratio\_value, 196  
 scale\_amp\_water\_ratio, 197  
 scale\_basis\_amp, 197  
 scale\_basis\_from\_singlet, 198  
 scale\_mrs\_amp, 198  
 scale\_spec, 199  
  
 sd(sd.list), 199  
 sd.list, 199  
 sd.mrs\_data, 200  
 seconds, 201  
 segment\_t1\_ants, 201  
 segment\_t1\_fsl, 202  
 segment\_t1\_rpyants, 202  
 seq\_cpmg\_ideal, 203  
 seq\_mega\_press\_ideal, 203  
 seq\_press\_2d\_shaped, 204  
 seq\_press\_ideal, 205  
 seq\_pulse\_acquire, 206  
 seq\_slaser\_ideal, 206  
 seq\_spin\_echo\_ideal, 207  
 seq\_steam\_ideal, 207  
 seq\_steam\_ideal\_cof, 208  
 seq\_steam\_ideal\_young, 208  
 set\_ants\_dir, 209  
 set\_def\_acq\_paras, 210  
 set\_lcm\_cmd, 210  
 set\_lw, 211  
 set\_mask\_xy\_mat, 211  
 set\_Ntrans, 212  
 set\_precomp\_mode, 212  
 set\_precomp\_verbose, 212  
 set\_ref, 213  
 set\_td\_pts, 213  
 set\_tqn\_cmd, 214  
 set\_tr, 214  
 shift, 215  
 shift\_basis, 215  
 sim\_asy\_pvoigt, 216  
 sim\_basis, 216  
 sim\_basis\_1h\_brain, 217  
 sim\_basis\_1h\_brain\_press, 218  
 sim\_basis\_mm\_lip\_lcm, 219  
 sim\_basis\_tqn, 219  
 sim\_brain\_1h, 220  
 sim\_mol, 221  
 sim\_noise, 222  
 sim\_resonances, 223  
 sim\_th\_excit\_profile, 224  
 sim\_zero, 224  
 smooth\_dyns, 225  
 sort\_basis, 226  
 spant (spant-package), 10  
 spant-package, 10  
 spant\_abfit\_benchmark, 226

spant\_sim\_fmrs\_dataset, 227  
spant\_simulation\_benchmark, 227  
spec\_decomp, 228  
spec\_op, 228  
spin\_sys, 229  
spm\_pve2categorical, 230  
ssp, 230  
stackplot, 231  
stackplot.fit\_result, 231  
stackplot.mrs\_data, 233  
sub\_even\_from\_odd\_dyns, 235  
sub\_first\_dyn, 236  
sub\_mean\_dyns, 236  
sub\_median\_dyns, 237  
sub\_odd\_from\_even\_dyns, 237  
subtract\_rest\_task, 235  
sum\_coils, 238  
sum\_dyns, 238  
sum\_mrs, 239  
sum\_mrs\_list, 239  
sv\_res\_table, 244  
svs\_1h\_brain\_analysis, 240  
svs\_1h\_brain\_analysis\_dev, 241  
svs\_1h\_brain\_batch\_analysis, 243  
  
t\_test\_spec, 247  
td2fd, 244  
td\_conv\_filt, 245  
tdsr, 245  
te, 246  
tr, 246  
trim\_paths, 247  
  
varpro\_3\_para\_opts, 248  
varpro\_basic\_opts, 248  
varpro\_opts, 249  
vec2mrs\_data, 250  
  
write\_basis, 251  
write\_basis\_niidir, 251  
write\_basis\_tqn, 252  
write\_mrs, 252  
write\_mrs\_nifti, 253  
write\_pulse\_ascii, 253  
  
zero\_fade\_spec, 254  
zero\_higher\_orders, 254  
zero\_td\_pts\_end, 255  
zf, 255  
zf\_xy, 256