# Package: fitnmr (via r-universe)

May 13, 2026

**Version** 1.1

**Title** Multidimensional Nuclear Magnetic Resonance Peak Fitting and Analysis

**Description** Tools for fitting and analyzing 1D-4D nuclear magnetic resonance spectra with analytical models of peak shapes and peak groups. The package reads spectra in 'NMRPipe' format, builds constrained parameter structures for chemical shifts, line widths, scalar couplings, volumes, and phases, and performs nonlinear least-squares optimization for iterative peak discovery or simultaneous fits across multiple spectra. It also provides methods for visualization, preprocessing, and kinetic analysis of 1D time-series data, including automated phase optimization, solvent suppression, time-domain correction for frequency shifts and line broadening, modeling spectra as linear combinations of two component spectra, and exponential rate fitting.

**License** GPL-3

**URL** https://github.com/smith-group/fitnmr/, https://smith-group.github.io/fitnmr/

**Imports** minpack.lm, abind, Matrix, methods, R6

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Suggests** knitr, rmarkdown, bookdown, gslnls, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Repository** https://smith-group.r-universe.dev

**Date/Publication** 2026-05-13 15:17:15 UTC

**RemoteUrl** https://github.com/smith-group/fitnmr

**RemoteRef** HEAD

**RemoteSha** 3e9bb7d758c8b5b9f00b77d3bcd579088475466c

# Contents

---

`fitnmr-package` *fitnmr Package Overview*

---

## Description

The functionality provided by the fitnmr package can be divided into several categories:

### Spectrum reading, plotting, and analysis

Spectra in the NMRPipe file format can be read with `read_nmrpipe`. Those spectra can be plotted with `contour_pipe`. The noise level within a spectrum (or any numeric vector) can be calculated with `noise_estimate`.

### Low-level fitting and plotting

The core fitting procedure consists of a two-step process: First, the fit input is created with `make_fit_input`. Second, the fit is executed with `perform_fit`. Prior to running the fit, constraints can be added to the fit with either `update_fit_bounds` or `limit_omega0_by_r2`.

Before or after a fit has been performed, you can extract the raw, starting, or fit spectral intensities with `get_spec_int`. Furthermore, there are convenience plotting functions for plotting 1D (`plot_fit_1d`) or 2D (`plot_fit_2d`) fits.

The `make_fit_input` function takes many different parameters. To make a new fit from an existing fit, possibly with a different set of spectra or otherwise modified fitting parameters, `param_list_to_arg_list` can be helpful in generating a list of parameters for `make_fit_input`.

### Modifying parameter lists for fits

In fitnmr, a "parameter list" is a named list data structure that has all the information necessary to describe a set of peaks and interdependencies between the parameters for those peaks. It contains three elements: `start_list` (or also `fit_list` after a fit has been performed), `group_list`, and `comb_list`. Fitting constraints can also be stored in `lower_list` and `upper_list`. Each of those is itself a named list of arrays corresponding to different parameters, namely `omega0`, `r2`, `m0`, `p0`, `p1`, and `omega0_comb`. To help manage the values stored in those lists (particularly `omega0` and `omega0_comb`), there are several convenience functions to select particular subsets, including `omega0_param_idx` and `coupling_param_idx`. Once a subset is made, the parameters can be read or changed using `param_values`.

### High-level fitting and plotting

`fit_peak_iter`

`param_list_to_peak_df`, `plot_peak_df`, `peak_df_to_param_list`, `peak_df_to_fit_input`

`fit_peak_cluster`, `fit_footprint`

`fit_peaks`, `make_param_list`

`spec_overlap_mat`

### NMRPipe simulation and processing

`ppm_to_pts`, `whz_to_pts`, `write_nmrdraw_peak_tab`, `sim_time_nd`, `nmr_pipe`

### Postprocessing and assignment

`height_assign`, `read_nmrdraw_peak_tab`

### Deprecated

`extract_params`, `get_spec_peak_int`

### Author(s)

**Maintainer**: Colin Smith <colin.smith@wesleyan.edu>

### See Also

Useful links:

- https://github.com/smith-group/fitnmr/
- https://smith-group.github.io/fitnmr/

## abind_list *Combine multi-dimensional arrays with lists*

### Description

Combine multi-dimensional arrays with lists

### Usage

```
abind_list(..., rev.along = NULL)
```

### Arguments

| | |
|---|---|
| ... | any number of lists with dimensions or a single list of list arrrays |
| rev.along | dimension to reverse before binding (passed to abind) |

## collapse_na *Collapse strings of repeated NAs in a vector with numeric names*

### Description

Collapse strings of repeated NAs in a vector with numeric names

### Usage

```
collapse_na(x)
```

### Arguments

| | |
|---|---|
| x | 1D array with dimnames having numeric positional values |

## collapse_na_array *Collapse blocks of NAs in an array with numeric dimension names*

### Description

Collapse blocks of NAs in an array with numeric dimension names

### Usage

```
collapse_na_array(x)
```

### Arguments

| | |
|---|---|
| x | array with names having numeric positional values |

`comb_vec_to_param_array` *Determine array of destination parameters from a source vector*

## Description

Determine array of destination parameters from a source vector

## Usage

```
comb_vec_to_param_array(
  comb_vec,
  comb_array,
  param_array = NULL,
  na_only = FALSE
)
```

## Arguments

| | |
|---|---|
| `comb_vec` | vector of source parameters |
| `comb_array` | array of 2xN data frames with mapping between vector and array |
| `param_array` | starting array of destination parameters |
| `na_only` | only overwrite values in param_array if they are NA |

`contour_pipe` *Plot Spectra Contours*

## Description

Plot a two dimensional contour plot

## Usage

```
contour_pipe(
  data_mat,
  nlevels = 10,
  zlim = range(data_mat, na.rm = TRUE),
  low_frac = 0.05,
  lwd = 0.25,
  main = NA,
  col_pos = "black",
  col_neg = grDevices::rgb(t(grDevices::col2rgb(col_pos))/255 * 0.25 + 0.75),
  add = FALSE,
  xlab = NULL,
  ylab = NULL,
  frame.plot = TRUE
)
```

## Arguments

| | |
|---|---|
| data_mat | matrix with spectral intensities. |
| nlevels | number of contour levels. |
| zlim | minimum and maximum values at which to show contours. |
| low_frac | minimum absolute value (as a fraction of zlim) at which to show contours. |
| lwd | width of contour lines. |
| main | title of the plot. |
| col_pos | color of positive contours. |
| col_neg | color of negative contours, defaults to lighter version of col_pos. |
| add | logical indicating whether to add to an existing plot (i.e. not start a new one). |
| xlab | label for x-axis, defaults to names(dimnames(datamat))[1]. |
| ylab | label for y-axis, defaults to names(dimnames(datamat))[2]. |
| frame.plot | a logical indicating whether a box should be drawn around the plot. |

## Details

The first dimension of data_matrix is drawn along the x-axis and the second dimension is drawn along the y-axis.

If low_frac is specified, then there can actually be up to nlevels total positive contour levels and/or nlevels total negative contour levels, whichever has the larger magnitude in zlim. The other dimension will have the mirror image of those up to the relevant limit in zlim. Note that the levels do not actually go up to zlim. There are nlevels+1 log spaced levels calculated from the contour determined by low_frac up to the maximum absolute zlim, but the final level is not drawn because it is at the maximum absolute zlim.

If low_frac is set to NA, then the contour levels are calculated in the same way as contour, with there being approximately nlevels linearly spaced levels in a range close to zlim.

Note that this function does not directly take the data returned by read_nmrpipe. You must pass the int matrix from the value returned by that function.

## Value

No return value, called for side effects (draws contour lines).

## Examples

```
spec_file <- system.file("extdata", "t1", "1.ft2", package = "fitnmr")
spec <- read_nmrpipe(spec_file, dim_order = "hx")
contour_pipe(spec$int)
```

## coupling_param_idx *Get a list of logical arrays indicating which parameters correspond to scalar couplings*

### Description

Get a list of logical arrays indicating which parameters correspond to scalar couplings

### Usage

```
coupling_param_idx(param_list, comb_idx_offset = 0)
```

### Arguments

| | |
|---|---|
| param_list | parameter list |
| comb_idx_offset | offset for combination index mapping |

### Value

A named ‘list‘ of logical index arrays/vectors (matching ‘param_list$group_list‘) that identify scalar-coupling parameters in ‘omega0_comb‘.

## extract_params *Extract parameters from fit object for use with make_fit_input*

### Description

Extract parameters from fit object for use with make_fit_input

### Usage

```
extract_params(fit, expand = 0)
```

### Arguments

| | |
|---|---|
| fit | fit_output structure |
| expand | number of empty peaks to append |

## fit_footprint — *Determine the region of a spectrum containing the majority of the fit peaks*

### Description

Determine the region of a spectrum containing the majority of the fit peaks

### Usage

```
fit_footprint(fit, frac = 0.12)
```

### Arguments

| | |
|---|---|
| fit | fit_output structure |
| frac | fraction of total intensity used to define the footprint |

## fit_peak_cluster — *Fit a cluster of nearby peaks starting from a seed table of chemical shifts*

### Description

Fit a cluster of nearby peaks starting from a seed table of chemical shifts

### Usage

```
fit_peak_cluster(
  spec_list,
  cs_start,
  spec_ord,
  f_alpha_thresh = 0.001,
  omega0_plus = c(0.075, 0.75),
  r2_start = 5,
  r2_bounds = c(0.5, 20),
  sc_start = NULL,
  sc_bounds = c(0, Inf),
  plot_main_prefix = NULL,
  peak_num_offset = 0,
  plot_fit_stages = FALSE,
  verbose = TRUE
)
```

### Arguments

| | |
|---|---|
| spec_list | list of spectra |
| cs_start | matrix of starting chemical shifts |
| spec_ord | order of spectral dimensions |
| f_alpha_thresh | alpha threshold for F-test |
| omega0_plus | omega0 bounds for each dimension |
| r2_start | starting r2 values |
| r2_bounds | numeric vector of length 2 with r2 bounds |
| sc_start | starting scalar coupling values |
| sc_bounds | numeric vector of length 2 with scalar coupling bounds |
| plot_main_prefix | optional prefix for plot titles |
| peak_num_offset | offset added to peak numbers in plots |
| plot_fit_stages | logical indicating whether to plot intermediate fits |
| verbose | logical indicating whether to print progress updates |

### Value

Usually a fit-output ‘list‘ (including ‘fit_list‘) for the accepted peak cluster. In edge cases where no acceptable fit is retained, returns modeled intensity matrix/matrices used for subtraction.

---

## fit_peak_iter *Iterative Peak Fitting*

---

### Description

Iteratively fit peaks for whole spectra

### Usage

```
fit_peak_iter(
  spectra,
  noise_sigma = NULL,
  noise_cutoff = 15,
  f_alpha = 0.001,
  iter_max = 100,
  omega0_plus = c(0.075, 0.75),
  r2_start = 5,
  r2_bounds = c(0.5, 20),
  sc_start = c(6, NA),
  sc_bounds = c(2, 12),
```

```
  fit_list = list(),
  plot_fit = FALSE,
  plot_fit_stages = FALSE,
  iter_callback = NULL,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| spectra | list of spectrum objects read by read_nmrpipe. |
| noise_sigma | numeric vector of noise levels associated with each spectrum. If NULL, it is calculated with noise_estimate. |
| noise_cutoff | numeric value multiplied by noise_sigma to determine cutoffs for each spectrum. Peak fitting will terminate if the maximum residuals for all spectra fall below these cutoffs. |
| f_alpha | numeric value giving a F-test p-value threshold above which a peak will not be accepted. |
| iter_max | integer maximum number of iterations to apply. |
| omega0_plus | numeric vector giving the window size (ppm plus or minus the starting omega0 values) around which to use points from the spectra for fitting. |
| r2_start | numeric vector giving the starting r2 value(s) for the fit (in Hz). |
| r2_bounds | numeric vector of length two giving the lower and upper bounds for r2. |
| sc_start | numeric vector giving the starting scalar coupling values for doublets. It should be the same length as the number of dimensions in the spectrum. Set the value to NA for a given dimension to make it a singlet. |
| sc_bounds | numeric vector of length two giving the lower and upper bounds for scalar couplings. |
| fit_list | list of previous fits to which the new fits should be appended. |
| plot_fit | logical indicating whether produce a fit cluster plot for each iteration. |
| plot_fit_stages | logical indicating whether to plot each stage of fitting within the iterations. |
| iter_callback | function called after each iteration with two arguments: iter and iter_max |
| verbose | logical indicating whether to print progress updates |

## Details

This function uses an iterative algorithm to fit all the peaks in a given list of spectra, assuming identical peak positions and shapes across the spectra. Each iteration starts by identifying the maximum value across all spectra, using that position to fit a cluster of overlapping peaks with the fit_peak_cluster function. After the cluster of peaks is fit, the modeled intensity is subtracted from all spectra and another iteration is performed. Iterations are terminated if max_iter is reached or the residual intensity in all spectra falls below noise_sigma*noise_cutoff.

This function currently only supports fitting of 2D spectra, but will be generalized to work with spectra of any dimensionality in the near future. To reduce the number of false positives/negatives,

the most important parameters to adjust are `noise_cutoff`, `f_alpha`, and `iter_max`. If `iter_max` is reached before all peaks have been identified, then you can call this function again, setting the `fit_list` parameter to the return value of the previous invocation. In that case, `iter_max` new iterations will be performed and appended to `fit_list`.

For visualizing the iterative algorithm as it progresses, you can enable either the `plot_fit` or `plot_fit_stages` parameters.

## Value

List of fit objects returned by `fit_peak_cluster`, one for each iteration. They are appended to `fit_list` if supplied.

## Examples

```
spec_file <- system.file("extdata", "t1", "1.ft2", package = "fitnmr")
spec <- list("1.ft2" = read_nmrpipe(spec_file, dim_order = "hx"))
peak_fits <- fit_peak_iter(spec, iter_max = 2)
```

---

# fit_peaks *Fit peaks from a table of chemical shifts*

---

## Description

Fit peaks from a table of chemical shifts

## Usage

```
fit_peaks(
  spec_list,
  cs_mat,
  fit_prev = NULL,
  spec_ord = 1:2,
  omega0_plus = c(0.075, 0.75),
  r2_start = 5,
  r2_bounds = c(0.5, 20),
  sc_start = NULL,
  sc_bounds = c(0, Inf),
  positive_only = TRUE,
  plot_fit_stages = TRUE
)
```

## Arguments

| | |
|---|---|
| `spec_list` | list of spectra |
| `cs_mat` | matrix of chemical shifts |
| `fit_prev` | optional previous fit parameters |

| | |
|---|---|
| spec_ord | order of spectral dimensions |
| omega0_plus | omega0 bounds for each dimension |
| r2_start | starting r2 values |
| r2_bounds | numeric vector of length 2 with r2 bounds |
| sc_start | starting scalar coupling values |
| sc_bounds | numeric vector of length 2 with scalar coupling bounds |
| positive_only | logical indicating whether to constrain m0 to be positive |
| plot_fit_stages | logical indicating whether to plot intermediate fits |

## Value

A fitted parameter 'list' (fit-output structure) containing fitted values in 'fit_list' plus the original fit metadata and bounds.

---

# get_spec_int *Get arrays of spectral intensities for input, starting parameters, and fit peaks*

---

## Description

Get arrays of spectral intensities for input, starting parameters, and fit peaks

## Usage

```
get_spec_int(
  fit_data,
  spec_type = c("input", "start", "fit"),
  spec_idx = seq_along(fit_data$spec_data),
  peak_idx = seq_len(dim(fit_data$start_list$omega0)[2])
)
```

## Arguments

| | |
|---|---|
| fit_data | fit_input or fit_output structure |
| spec_type | character indicating which spectra to return |
| spec_idx | indices of spectra to return |
| peak_idx | indices of peaks to include (for start/fit) |

## Value

A 'list' of numeric arrays, one per selected spectrum. Each array is on contiguous ppm axes and contains either observed ('input') or modeled ('start'/'fit') intensities.

## Examples

```
spec_file <- system.file("extdata", "t1", "1.ft2", package = "fitnmr")
spec <- read_nmrpipe(spec_file, dim_order = "hx")
fit_input <- make_fit_input(
  list(spec),
  omega0_start = matrix(c(8.5400, 119.76), nrow = 2),
  omega0_plus = c(0.075, 0.75),
  r2_start = 4,
  m0_start = 1e9
)
input_int <- get_spec_int(fit_input, "input")
start_int <- get_spec_int(fit_input, "start")
```

---

# get_spec_peak_int *Get spectra for individual peaks*

---

## Description

Get spectra for individual peaks

## Usage

```
get_spec_peak_int(
  fit_data,
  spec_type = c("start", "fit"),
  spec_idx = seq_along(fit_data$spec_data),
  peak_idx = seq_len(dim(fit_data$start_list$omega0)[2])
)
```

## Arguments

| | |
|---|---|
| fit_data | fit_input or fit_output structure |
| spec_type | character indicating which spectra to return |
| spec_idx | indices of spectra to return |
| peak_idx | indices of peaks to include |

`HamiltonianMultiplet` *HamiltonianMultiplet R6 class*

## Description

HamiltonianMultiplet R6 class

HamiltonianMultiplet R6 class

## Details

Build and differentiate NMR Hamiltonians for fixed-size spin systems.

## Public fields

`n` Number of spins in the system.

`shift_labels` Character vector of shift labels (diagonal of label_matrix).

`coupling_labels` Matrix of coupling labels (off-diagonal of label_matrix).

`params` Named numeric vector of parameter values.

`ops` List of spin operators.

`dH` List of Hamiltonian derivative matrices by parameter label.

`cache` Internal cache of eigendecomposition results.

## Methods

### Public methods:

- `HamiltonianMultiplet$new()`
- `HamiltonianMultiplet$set_params()`
- `HamiltonianMultiplet$get_param_labels()`
- `HamiltonianMultiplet$hamiltonian()`
- `HamiltonianMultiplet$operator()`
- `HamiltonianMultiplet$multiplet()`
- `HamiltonianMultiplet$clone()`

**Method** `new()`**:** Create a HamiltonianMultiplet from labels and parameters.

*Usage:*

```
HamiltonianMultiplet$new(label_matrix, params = NULL)
```

*Arguments:*

`label_matrix` Square matrix or data.frame of shift (diagonal) and coupling labels.

`params` Named numeric vector or list of parameter values.

*Returns:* A new HamiltonianMultiplet object.

**Method** `set_params()`**:** Update parameter values and mark the cache dirty if changed.

*Usage:*

```
HamiltonianMultiplet$set_params(params)
```

*Arguments:*

`params` Named numeric vector or list of parameter values.

**Method** `get_param_labels()`**:** Return sorted unique parameter labels.

*Usage:*

```
HamiltonianMultiplet$get_param_labels()
```

**Method** `hamiltonian()`**:** Return the Hamiltonian matrix for current parameters.

*Usage:*

```
HamiltonianMultiplet$hamiltonian()
```

**Method** `operator()`**:** Return the operator matrix for a label expression.

*Usage:*

```
HamiltonianMultiplet$operator(label)
```

*Arguments:*

`label` Operator label string.

**Method** `multiplet()`**:** Return a data.frame of transitions, intensities, and derivatives.

*Usage:*

```
HamiltonianMultiplet$multiplet(
  state_label,
  detect_label = "allx",
  intensity_tol = 0.1,
  coherence = 1,
  coherence_tol = 1e-06,
  normalize = TRUE
)
```

*Arguments:*

`state_label` Operator label string for the prepared state.

`detect_label` Operator label string for detection. Defaults to `"allx"`.

`intensity_tol` Minimum absolute intensity to keep in the output.

`coherence` Allowed coherence order(s) (integer vector) or NULL for all.

`coherence_tol` Tolerance for matching coherence orders.

`normalize` Logical; if TRUE, normalize intensities.

**Method** `clone()`**:** The objects of this class are cloneable with this method.

*Usage:*

```
HamiltonianMultiplet$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## height_assign *Calculate mapping from assigned peak list onto an unknown peak list*

### Description

This uses a greedy algorithm. It iterates over the unknown peaks in order of decreasing height and assigns each unknown to the closest assigned peak within a distance determined by the thresh parameter, as long as that peak wasn't already assigned.

### Usage

```
height_assign(assigned, unknown, thresh = 0.01)
```

### Arguments

| | |
|---|---|
| assigned | two column matrix with assigned peak chemical shifts |
| unknown | three column matrix with unknown peak chemical shifts and heights |
| thresh | maximum distance (as a fraction of the two chemical shift ranges) |

### Value

An integer vector of length 'nrow(assigned)' giving matched row indices in 'unknown' (or 'NA' when unmatched). The 'thresh' attribute stores the effective distance thresholds in each dimension.

## infer_acquisition_time *Infer acquisition time for each dimension*

### Description

Infer acquisition time for each dimension

### Usage

```
infer_acquisition_time(fheader, empirically_correct = TRUE)
```

### Arguments

| | |
|---|---|
| fheader | matrix of generalized ND parameters |
| empirically_correct | logical indicating whether to apply empirical correction |

## infer_aliasing *Infer which dimension was directly acquired*

### Description

Infer which dimension was directly acquired

### Usage

```
infer_aliasing(fheader, phase_tolerance = 10)
```

### Arguments

| | |
|---|---|
| fheader | matrix of generalized ND parameters |
| phase_tolerance | tolerance in degrees for detecting half-dwell delay |

## infer_direct *Infer which dimension was directly acquired*

### Description

Infer which dimension was directly acquired

### Usage

```
infer_direct(fheader)
```

### Arguments

| | |
|---|---|
| fheader | matrix of generalized ND parameters |

## infer_sweep_width *Infer original sweep width for each dimension*

### Description

Infer original sweep width for each dimension

### Usage

```
infer_sweep_width(fheader)
```

### Arguments

| | |
|---|---|
| fheader | matrix of generalized ND parameters |

## limit_omega0_by_r2 *Add upper/lower limits based on the r2 value*

### Description

Add upper/lower limits based on the r2 value

### Usage

```
limit_omega0_by_r2(fit_input, factor = 1.5)
```

### Arguments

| | |
|---|---|
| fit_input | fit_input structure |
| factor | multiplier applied to r2 to set omega0 bounds |

### Value

A modified ‘fit_input‘ list with updated ‘lower_list$omega0‘ and ‘upper_list$omega0‘ bounds.

### Examples

```
spec_file <- system.file("extdata", "t1", "1.ft2", package = "fitnmr")
spec <- read_nmrpipe(spec_file, dim_order = "hx")
fit_input <- make_fit_input(
  list(spec),
  omega0_start = matrix(c(8.5400, 119.76), nrow = 2),
  omega0_plus = c(0.075, 0.75),
  r2_start = 4,
  m0_start = 1e9
)
fit_input$start_list[1:2]
fit_input$lower_list[1]
fit_input$upper_list[1]
fit_input <- limit_omega0_by_r2(fit_input)
fit_input$lower_list[1]
fit_input$upper_list[1]
```

## make_coupling_mat *Make a multiplet matrix with weights and scalar coupling coefficients*

### Description

Make a multiplet matrix with weights and scalar coupling coefficients

### Usage

```
make_coupling_mat(sc_names)
```

### Arguments

| | |
|---|---|
| sc_names | character vector with names of scalar couplings |

---

## make_fit_input *Prepare input data structure for peak fitting*

---

### Description

Prepare input data structure for peak fitting

### Usage

```
make_fit_input(
  spectra,
  omega0_start,
  omega0_plus,
  omega0_minus = omega0_plus,
  omega0_trunc = NULL,
  r2_start = NULL,
  m0_start = NULL,
  m0_region = (omega0_plus + omega0_minus)/2,
  p0_start = 0,
  p1_start = 0,
  omega0_group = NULL,
  r2_group = NULL,
  m0_group = NULL,
  p0_group = 0,
  p1_group = 0,
  omega0_comb = NULL,
  omega0_comb_start = NULL,
  omega0_comb_group = NULL,
  coupling_comb = NULL,
  spinsystems = NULL,
  couplings = NULL,
  resonance_names = NULL,
  nucleus_names = NULL,
  field_offsets = numeric(),
  field_start = numeric(),
  field_group = 0,
  fheader = NULL
)
```

## Arguments

| | |
|---|---|
| spectra | list of spectra or data frame with ppm and intensity columns |
| omega0_start | starting omega0 values |
| omega0_plus | positive omega0 bounds |
| omega0_minus | negative omega0 bounds |
| omega0_trunc | optional truncation bounds for omega0 |
| r2_start | starting r2 values |
| m0_start | starting m0 values |
| m0_region | region for m0 estimation |
| p0_start | starting zero-order phase values |
| p1_start | starting first-order phase values |
| omega0_group | grouping for omega0 values |
| r2_group | grouping for r2 values |
| m0_group | grouping for m0 values |
| p0_group | grouping for p0 values |
| p1_group | grouping for p1 values |
| omega0_comb | list of omega0 combination matrices |
| omega0_comb_start | starting omega0 combination values |
| omega0_comb_group | grouping for omega0 combinations |
| coupling_comb | list of coupling combination matrices |
| spinsystems | optional named list of spin system label matrices |
| couplings | optional couplings table or named vector of coupling values |
| resonance_names | optional resonance names |
| nucleus_names | optional nucleus names |
| field_offsets | matrix of field offsets |
| field_start | starting field values |
| field_group | grouping for field values |
| fheader | optional fheader matrix when spectra is a data frame |

## Value

A ‘fit_input‘ ‘list‘ containing ‘spec_data‘, parameter lists (‘start_list‘, ‘group_list‘, ‘comb_list‘), parameter bounds (‘lower_list‘, ‘upper_list‘), naming metadata, and ‘num_points‘.

## Examples

```
spec_file <- system.file("extdata", "t1", "1.ft2", package = "fitnmr")
spec <- read_nmrpipe(spec_file, dim_order = "hx")
fit_input <- make_fit_input(
  list(spec),
  omega0_start = matrix(c(8.5400, 119.76), nrow = 2),
  omega0_plus = c(0.075, 0.75),
  r2_start = 4,
  m0_start = 1e9
)
str(fit_input)
```

---

# make_map *Create a sparse axis map*

---

## Description

Create a sparse axis map

## Usage

```
make_map(
  x,
  max_spacing = 0.125,
  new_spacing = 0.025,
  starts = NULL,
  ends = NULL
)
```

## Arguments

| | |
|---|---|
| x | numeric vector or array used to define the sparse axis |
| max_spacing | maximum spacing between segments before inserting a gap |
| new_spacing | spacing used for the inserted gap |
| starts | optional numeric vector of segment start positions |
| ends | optional numeric vector of segment end positions |

## Value

A two-column numeric matrix mapping original coordinates (column 1) to sparse-axis coordinates (column 2), with ‘starts‘ and ‘ends‘ attributes giving segment boundaries.

`make_param_list` *Make a parameter list for a set of spectra and chemical shifts*

## Description

Make a parameter list for a set of spectra and chemical shifts

## Usage

```
make_param_list(
  spec_list,
  cs_mat,
  fit_prev = NULL,
  r2_start = 5,
  m0_start = 1,
  sc_start = NULL,
  same_r2 = FALSE,
  same_coupling = FALSE
)
```

## Arguments

| | |
|---|---|
| `spec_list` | list of spectra |
| `cs_mat` | matrix of chemical shifts |
| `fit_prev` | optional previous fit parameters |
| `r2_start` | starting r2 values |
| `m0_start` | starting m0 values |
| `sc_start` | starting scalar coupling values |
| `same_r2` | logical indicating whether to tie r2 across dimensions |
| `same_coupling` | logical indicating whether to tie coupling across dimensions |

## Value

A parameter ‘list‘ with ‘start_list‘, ‘group_list‘, and ‘comb_list‘ components that can be passed to ‘make_fit_input()‘.

`nmr_pipe` *Process an FID with NMRPipe*

## Description

Process an FID with NMRPipe

## Usage

```
nmr_pipe(
  in_path,
  out_path,
  ndim = 1,
  apod = NULL,
  sp = rbind(off = 0.5, end = 1, pow = 1, c = 0.5),
  zf = rbind(auto = ""),
  ps = rbind(p0 = 0, p1 = 0, di = ""),
  ext = NULL
)
```

## Arguments

| | |
|---|---|
| `in_path` | input file path |
| `out_path` | output file path |
| `ndim` | number of dimensions |
| `apod` | optional apodization argument matrix |
| `sp` | optional sine-bell argument matrix |
| `zf` | optional zero-fill argument matrix |
| `ps` | optional phase correction argument matrix |
| `ext` | optional extraction argument matrix |

## Value

The integer exit status from ‘system(...)‘ (typically ‘0‘ on success).

## nmrpipe_ft *Fourier transform a 1D FID*

### Description

This function aims to numerically match the NMRPipe ‘FT‘ function.

### Usage

```
nmrpipe_ft(fid)
```

### Arguments

| | |
|---|---|
| fid | list with ‘int‘, ‘header‘, and ‘fheader‘ elements containing FID data |

### Details

See the NMRPipe documentation: https://www.nmrscience.com/ref/nmrpipe/ft.html

### Value

A transformed spectrum ‘list‘ with updated ‘int‘, ‘ppm‘, ‘header‘, and ‘fheader‘ fields.

### Examples

```
fid_path <- system.file("extdata", "noesy1d", "11", "test.fid", package = "fitnmr")
fid <- read_nmrpipe(fid_path, complex_data = TRUE)
sp <- nmrpipe_sp(fid)
zf <- nmrpipe_zf(sp)
ft <- nmrpipe_ft(zf)
```

## nmrpipe_fti *Inverse Fourier transform a 1D spectrum*

### Description

This function aims to numerically match the NMRPipe ‘FT -inv‘ function.

### Usage

```
nmrpipe_fti(ft)
```

### Arguments

| | |
|---|---|
| ft | list with ‘int‘, ‘header‘, and ‘fheader‘ elements containing spectrum data |

## Details

See the NMRPipe documentation: https://www.nmrscience.com/ref/nmrpipe/ft.html

## Value

A modified FID-like ‘list‘ with inverse-transformed ‘int‘ values and updated ‘header‘/‘fheader‘ metadata.

## Examples

```
fid_path <- system.file("extdata", "noesy1d", "11", "test.fid", package = "fitnmr")
fid <- read_nmrpipe(fid_path, complex_data = TRUE)
sp <- nmrpipe_sp(fid)
zf <- nmrpipe_zf(sp)
ft <- nmrpipe_ft(zf)
fti <- nmrpipe_fti(ft)
```

---

# nmrpipe_ps *Change phases of a 1D spectrum*

---

## Description

This function aims to numerically match the NMRPipe ‘PS‘ function.

## Usage

```
nmrpipe_ps(ft, p0 = 0, p1 = 0)
```

## Arguments

| | |
|---|---|
| ft | list with ‘int‘, ‘header‘, and ‘fheader‘ elements containing spectrum data |
| p0 | equivalent to ‘-p0‘ flag |
| p1 | equivalent to ‘-p1‘ flag |

## Details

See the NMRPipe documentation: https://www.nmrscience.com/ref/nmrpipe/ps.html

## Value

A modified spectrum ‘list‘ with phase-corrected ‘int‘ values and updated phase metadata in ‘header‘ and ‘fheader‘.

## Examples

```
fid_path <- system.file("extdata", "noesy1d", "11", "test.fid", package = "fitnmr")
fid <- read_nmrpipe(fid_path, complex_data = TRUE)
sp <- nmrpipe_sp(fid)
zf <- nmrpipe_zf(sp)
ft <- nmrpipe_ft(zf)
ps <- nmrpipe_ps(ft, p0 = 129, p1 = 3)
```

---

`nmrpipe_sp` *Apply sine-based window function to a 1D FID*

---

## Description

This function aims to numerically match the NMRPipe ‘SP‘ function.

## Usage

```
nmrpipe_sp(fid, off = 0, end = 1, pow = 1, cval = 1, dmx = FALSE)
```

## Arguments

| | |
|---|---|
| `fid` | list with ‘int‘, ‘header‘, and ‘fheader‘ elements containing FID data |
| `off` | equivalent to ‘-off‘ flag |
| `end` | equivalent to ‘-end‘ flag |
| `pow` | equivalent to ‘-pow‘ flag |
| `cval` | equivalent to ‘-c‘ flag |
| `dmx` | equivalent to ‘-dmx‘ flag |

## Details

See the NMRPipe documentation: https://www.nmrscience.com/ref/nmrpipe/sp.html

## Value

A modified FID ‘list‘ with updated ‘int‘, ‘header‘, and ‘fheader‘ values after sine-bell apodization.

## Examples

```
fid_path <- system.file("extdata", "noesy1d", "11", "test.fid", package = "fitnmr")
fid <- read_nmrpipe(fid_path, complex_data = TRUE)
sp <- nmrpipe_sp(fid)
```

## nmrpipe_zf *Apply zero filling to a 1D FID*

### Description

This function aims to numerically match the NMRPipe ‘ZF‘ function.

### Usage

```
nmrpipe_zf(fid, zf = 1, pad = NULL, size = NULL, auto = FALSE)
```

### Arguments

| | |
|---|---|
| fid | list with ‘int‘, ‘header‘, and ‘fheader‘ elements containing FID data |
| zf | equivalent to ‘-zf‘ flag |
| pad | equivalent to ‘-pad‘ flag |
| size | equivalent to ‘-size‘ flag |
| auto | equivalent to ‘-auto‘ flag |

### Details

See the NMRPipe documentation: https://www.nmrscience.com/ref/nmrpipe/zf.html

### Value

A modified FID ‘list‘ with zero-filled ‘int‘ data and updated ‘header‘/‘fheader‘ metadata.

### Examples

```
fid_path <- system.file("extdata", "noesy1d", "11", "test.fid", package = "fitnmr")
fid <- read_nmrpipe(fid_path, complex_data = TRUE)
sp <- nmrpipe_sp(fid)
zf <- nmrpipe_zf(sp)
```

## noise_estimate *Estimate Noise*

### Description

Estimate properties of noise by fitting a Gaussian to a histogram of intensities

## Usage

```
noise_estimate(
  x,
  height = TRUE,
  thresh = 10,
  plot_distributions = TRUE,
  peak_intensities = NULL
)
```

## Arguments

| | |
|---|---|
| x | numeric values for which to estimate noise. |
| height | logical indicating whether to use Gaussian function whose area is not fixed at one because of an additional height scaling factor. |
| thresh | numeric value specifying the factor by which to multiply the standard deviation to determine the threshold away from the mean value within which to include values for the next iteration and final histogram for fitting. |
| plot_distributions | logical indicating whether to plot the distribution and Gaussian fit used to estimate the noise. |
| peak_intensities | numeric values of peak intensities to determine the signal to noise. |

## Details

This function estimates noise using iterative calculation of the mean and standard deviation, followed by fitting a Gaussian function to a histogram of the values within a range determined at the end of the iterations.

The iterative algorithm first calculates the mean and standard deviation of the values in `x`. In the next iteration, only points within `thresh` times the standard deviation of the mean value are used for calculating a new mean and standard deviation. This is repeated 20 times. A histogram with 512 bins is then computed within a range determined from the final mean and standard deviation. The returned parameters are based on a Gaussian fit to this histogram.

## Value

a named numeric vector with values:

**mean** mean value from the Gaussian fit

**mu** standard deviation from the Gaussian fit

**h** height of Gaussian fit (optional depending on value of `height` parameter)

**max** maximum value of `x`

## Examples

```
spec_file <- system.file("extdata", "t1", "1.ft2", package="fitnmr")
spec <- read_nmrpipe(spec_file, dim_order="hx")
noise_estimate(spec$int)
```

---

`omega0_comb_source_idx` *Get the first index in the omega0 array corresponding to each TRUE value in omega0_idx*

---

## Description

Get the first index in the omega0 array corresponding to each TRUE value in omega0_idx

## Usage

```
omega0_comb_source_idx(param_list, omega0_idx = omega0_param_idx(param_list))
```

## Arguments

| | |
|---|---|
| `param_list` | parameter list |
| `omega0_idx` | logical index list for omega0 parameters |

---

`omega0_param_idx` *Get a list of logical arrays indicating which parameters correspond to peak positions*

---

## Description

Get a list of logical arrays indicating which parameters correspond to peak positions

## Usage

```
omega0_param_idx(
  param_list,
  dims = seq_len(dim(param_list[["group_list"]][["omega0"]])[1]),
  peaks = seq_len(dim(param_list[["group_list"]][["omega0"]])[2]),
  specs = seq_len(dim(param_list[["group_list"]][["omega0"]])[3])
)
```

## Arguments

| | |
|---|---|
| `param_list` | parameter list |
| `dims` | dimensions to include |
| `peaks` | peaks to include |
| `specs` | spectra to include |

### Value

A named ‘list‘ of logical index arrays/vectors (matching ‘param_list$group_list‘) that identify omega0-related parameters to extract or modify.

---

`param_array_to_comb_vec` *Determine vector of source parameters from destination array via least squares*

---

### Description

Determine vector of source parameters from destination array via least squares

### Usage

```
param_array_to_comb_vec(
  param_array,
  comb_array,
  group_vec = NULL,
  resid_thresh = 1e-08
)
```

### Arguments

| | |
|---|---|
| `param_array` | array of destination parameters |
| `comb_array` | array of 2xN data frames with mapping between vector and array |
| `group_vec` | optional vector of group numbers for source parameters |
| `resid_thresh` | all residuals from least squares fit must be less than this value |

---

`param_list_to_arg_list` *Convert a list of parameters for use with make_fit_input*

---

### Description

Convert a list of parameters for use with make_fit_input

### Usage

```
param_list_to_arg_list(param_list)
```

### Arguments

| | |
|---|---|
| `param_list` | parameter list |

## Value

A flattened, named ‘list‘ of arguments suitable for ‘do.call()‘ with ‘make_fit_input()‘.

---

`param_list_to_peak_df` *Convert Fit to Data Frame*

---

## Description

Convert a parameter list into a peak data frame

## Usage

```
param_list_to_peak_df(param_list, spec_names = NULL)
```

## Arguments

| | |
|---|---|
| `param_list` | list of fit parameters (or a list of such lists) |
| `spec_names` | character vector of spectrum names |

## Details

This function takes the input or (if present) output parameters from a fit and converts them into a data frame. A parameter list must contain three lists:

`start_list` **or** `fit_list` input or output values of the respective fit parameters

`group_list` group numbers for the fit parameters

`comb_list` coefficients for deriving fit parameters from a linear combination other auxiliary parameters

This function currently assumes the fit parameters were generated by `fit_peak_iter`, `fit_peak_cluster`, or `peak_df_to_param_list`. These functions use a particular convention for `group_list` and `comb_list` to represent either singlets or doublets in each dimension of a 2D spectrum.

This function can take either a single parameter list or a list of parameter lists. If the latter is given, then the results from all the parameter lists will be combined into a single table.

## Value

A data frame with the following columns:

`peak` peak number

`fit` fit cluster number, with all peaks in the same cluster having the same `r2`, scalar couplings, and `m0`

`f_pvalue` optional, p-value determined from F-test during iterative fitting

`omega0_ppm_1` chemical shift of singlet/doublet center in the first dimension (ppm)

`omega0_ppm_2` chemical shift of singlet/doublet center in the second dimension (ppm)

sc_hz_1 optional, scalar coupling of doublet in first dimension (Hz)

sc_hz_2 optional, scalar coupling of doublet in second dimension (Hz)

r2_hz_1 R2 in first dimension (Hz)

r2_hz_2 R2 in first dimension (Hz)

... m0 values for each spectrum

## Examples

```
spec_file <- system.file("extdata", "t1", "1.ft2", package = "fitnmr")
spec <- list("1.ft2" = read_nmrpipe(spec_file, dim_order = "hx"))
peak_fits <- fit_peak_iter(spec, iter_max = 2)
param_list_to_peak_df(peak_fits)
```

---

# param_list_to_tables *Convert a parameter list into a set of tables with resonance/nuclei/couplings*

---

## Description

Convert a parameter list into a set of tables with resonance/nuclei/couplings

## Usage

```
param_list_to_tables(param_list, tables)
```

## Arguments

| | |
|---|---|
| param_list | param_list, fit_input, or fit_output structure |
| tables | list with resonance/nuclei/couplings tables to update |

## Value

A named ‘list‘ with updated ‘resonances‘, ‘nuclei‘, and ‘couplings‘ data frames.

## Examples

```
spec_file <- system.file("extdata", "tyrosine", "proton.ft1", package = "fitnmr")
spec <- read_nmrpipe(spec_file)

start_resonances <- structure(
  list(
    x = c("HA", "HB3", "HB2", "HD1/2", "HE1/2"),
    x_sc = c(
      "HA-HB3 HA-HB2",
      "HA-HB3 HB3-HB2",
      "HA-HB2 HB3-HB2",
      "HD1/2-HE1/2-3 HD1/2-HE1/2-5",
```

```
      "HD1/2-HE1/2-3 HD1/2-HE1/2-5"
    )
  ),
  row.names = c("HA", "HB3", "HB2", "HD1/2", "HE1/2"),
  class = "data.frame"
)

start_nuclei <- structure(
  list(
    omega0_ppm = c(3.364, 2.635, 2.799, 6.94, 6.538),
    r2_hz = c(0.7, 0.7, 0.7, 0.7, 0.7)
  ),
  class = "data.frame",
  row.names = c("HA", "HB3", "HB2", "HD1/2", "HE1/2")
)

start_couplings <- structure(
  list(hz = c(7.153, 5.159, -13.941, 7.7, 2)),
  class = "data.frame",
  row.names = c("HA-HB3", "HA-HB2", "HB3-HB2", "HD1/2-HE1/2-3", "HD1/2-HE1/2-5")
)

start_tables <- list(
  resonances = start_resonances,
  nuclei = start_nuclei,
  couplings = start_couplings
)
param_list <- tables_to_param_list(list(spec), start_tables)
param_list$start_list$m0[] <- 1e9
arg_list <- param_list_to_arg_list(param_list)
fit_input <- do.call(make_fit_input, c(list(list(spec), omega0_plus = 0.075), arg_list))
fit_output <- perform_fit(fit_input)
fit_tables <- param_list_to_tables(fit_output, start_tables)
```

---

`param_values` *Get/set a subset of fitting parameters specified by a list of logical vectors*

---

## Description

Get/set a subset of fitting parameters specified by a list of logical vectors

## Usage

```
param_values(params, idx_list)

param_values(params, idx_list) <- value
```

### Arguments

| | |
|---|---|
| params | a list of fitting parameters |
| idx_list | a list with the same structure as params but with logical vectors indicating which values should be return/set |
| value | replacement values (for setter) |

### Value

a vector of parameter values or the modified parameter list

---

## peak_bind *Combine parameter lists referring to different peaks*

---

### Description

Combine parameter lists referring to different peaks

### Usage

```
peak_bind(...)
```

### Arguments

| | |
|---|---|
| ... | any number of parameter lists or a single list of parameter lists |

---

## peak_df_to_fit_input *Convert a peak data frame to fit input*

---

### Description

Convert a peak data frame to fit input

### Usage

```
peak_df_to_fit_input(peak_df, spectra, ...)
```

### Arguments

| | |
|---|---|
| peak_df | data frame of peak parameters |
| spectra | list of spectra |
| ... | additional arguments passed to make_fit_input |

### Value

A ‘fit_input‘ list ready for optimization, as produced by ‘make_fit_input()‘.

## Examples

```
spec_file <- system.file("extdata", "t1", "1.ft2", package = "fitnmr")
spectra <- list("1.ft2" = read_nmrpipe(spec_file, dim_order = "hx"))

peak_df <- data.frame(
  peak = c(1, 2, 3, 4),
  fit = c(1, 1, 2, 2),
  f_pvalue = c(4.566421e-10, 1.118991e-05, 1.876528e-15, 5.817124e-04),
  omega0_ppm_1 = c(8.247602, 8.259565, 8.540030, 8.520232),
  omega0_ppm_2 = c(121.8666, 121.9299, 119.7611, 119.7266),
  sc_hz_1 = c(3.280589, 3.280589, 2.000000, 2.000000),
  r2_hz_1 = c(2.907218, 2.907218, 4.788566, 4.788566),
  r2_hz_2 = c(2.334497, 2.334497, 2.099646, 2.099646),
  `1.ft2` = c(824420657, 240560662, 1020008726, 89977216),
  check.names = FALSE
)

fit_input <- peak_df_to_fit_input(peak_df, spectra, omega0_plus=c(0.075, 0.75))
```

---

# peak_df_to_param_list *Convert a peak data frame to a parameter list*

---

## Description

Convert a peak data frame to a parameter list

## Usage

```
peak_df_to_param_list(peak_df, spectra)
```

## Arguments

| | |
|---|---|
| peak_df | data frame of peak parameters |
| spectra | list of spectra |

## Value

A parameter ‘list‘ with ‘start_list‘, ‘group_list‘, and ‘comb_list‘ inferred from ‘peak_df‘ and aligned to ‘spectra‘.

## Examples

```
spec_file <- system.file("extdata", "t1", "1.ft2", package = "fitnmr")
spectra <- list("1.ft2" = read_nmrpipe(spec_file, dim_order = "hx"))

peak_df <- data.frame(
  peak = c(1, 2, 3, 4),
  fit = c(1, 1, 2, 2),
```

```
  f_pvalue = c(4.566421e-10, 1.118991e-05, 1.876528e-15, 5.817124e-04),
  omega0_ppm_1 = c(8.247602, 8.259565, 8.540030, 8.520232),
  omega0_ppm_2 = c(121.8666, 121.9299, 119.7611, 119.7266),
  sc_hz_1 = c(3.280589, 3.280589, 2.000000, 2.000000),
  r2_hz_1 = c(2.907218, 2.907218, 4.788566, 4.788566),
  r2_hz_2 = c(2.334497, 2.334497, 2.099646, 2.099646),
  `1.ft2` = c(824420657, 240560662, 1020008726, 89977216),
  check.names = FALSE
)

peak_df_to_param_list(peak_df, spectra)
```

---

`perform_fit` *Perform a fit with an input data structure*

---

## Description

Perform a fit with an input data structure

## Usage

```
perform_fit(
  fit_input,
  method = c("minpack.lm", "gslnls", "sparseLM", "L-BFGS-B"),
  max_iter = 200,
  ...
)
```

## Arguments

| | |
|---|---|
| `fit_input` | fit_input structure |
| `method` | fitting method |
| `max_iter` | maximum number of iterations for method = "minpack.lm" |
| `...` | additional arguments passed to the fitting routine |

## Value

The input ‘fit_input‘ list, augmented with fitted parameters in ‘fit_list‘ and fit diagnostics in ‘fit_rsstrace‘, ‘fit_counts‘, and ‘fit_time‘.

## Examples

```
spec_file <- system.file("extdata", "t1", "1.ft2", package = "fitnmr")
spec <- read_nmrpipe(spec_file, dim_order = "hx")
fit_input <- make_fit_input(
  list(spec),
  omega0_start = matrix(c(8.5400, 119.76), nrow = 2),
```

```
  omega0_plus = c(0.075, 0.75),
  r2_start = 4,
  m0_start = 1e9
)
fit_output <- perform_fit(fit_input)
fit_output$start_list[1:3]
fit_output$fit_list[1:3]
```

---

`plot_fit_1d` *Plot a one dimensional peak fit*

---

## Description

Plot a one dimensional peak fit

## Usage

```
plot_fit_1d(fit_data, always_show_start = FALSE)
```

## Arguments

| | |
|---|---|
| `fit_data` | fit_input or fit_output structure |
| `always_show_start` | logical indicating whether to show starting model when fit is present |

## Value

No return value, called for side effects (draws plots).

## Examples

```
spec_file <- system.file("extdata", "tyrosine", "proton.ft1", package = "fitnmr")
spec <- read_nmrpipe(spec_file)

start_resonances <- structure(
  list(
    x = c("HA", "HB3", "HB2", "HD1/2", "HE1/2"),
    x_sc = c(
      "HA-HB3 HA-HB2",
      "HA-HB3 HB3-HB2",
      "HA-HB2 HB3-HB2",
      "HD1/2-HE1/2-3 HD1/2-HE1/2-5",
      "HD1/2-HE1/2-3 HD1/2-HE1/2-5"
    )
  ),
  row.names = c("HA", "HB3", "HB2", "HD1/2", "HE1/2"),
  class = "data.frame"
)
```

```
start_nuclei <- structure(
  list(
    omega0_ppm = c(3.364, 2.635, 2.799, 6.94, 6.538),
    r2_hz = c(0.7, 0.7, 0.7, 0.7, 0.7)
  ),
  class = "data.frame",
  row.names = c("HA", "HB3", "HB2", "HD1/2", "HE1/2")
)

start_couplings <- structure(
  list(hz = c(7.153, 5.159, -13.941, 7.7, 2)),
  class = "data.frame",
  row.names = c("HA-HB3", "HA-HB2", "HB3-HB2", "HD1/2-HE1/2-3", "HD1/2-HE1/2-5")
)

start_tables <- list(
  resonances = start_resonances,
  nuclei = start_nuclei,
  couplings = start_couplings
)
param_list <- tables_to_param_list(list(spec), start_tables)
param_list$start_list$m0[] <- 1e9
arg_list <- param_list_to_arg_list(param_list)
fit_input <- do.call(make_fit_input, c(list(list(spec), omega0_plus = 0.075), arg_list))
fit_output <- perform_fit(fit_input)
plot_fit_1d(fit_output)
```

---

`plot_fit_2d` *Plot a two dimensional peak fit*

---

## Description

Plot a two dimensional peak fit

## Usage

```
plot_fit_2d(
  fit_output,
  spec_ord = seq_len(dim(fit_output$start_list$omega0)[1]),
  always_show_start = FALSE,
  main = NULL
)
```

## Arguments

| | |
|---|---|
| `fit_output` | fit_output structure |
| `spec_ord` | order of spectral dimensions to plot |

always_show_start
: logical indicating whether to show starting model when fit is present

main
: optional title for the plot

## Value

No return value, called for side effects (draws plots).

## Examples

```
spec_file <- system.file("extdata", "t1", "1.ft2", package = "fitnmr")
spec <- read_nmrpipe(spec_file, dim_order = "hx")
fit_input <- make_fit_input(
  list(spec),
  omega0_start = matrix(c(8.5400, 119.76), nrow = 2),
  omega0_plus = c(0.075, 0.75),
  r2_start = 4,
  m0_start = 1e9
)
plot_fit_2d(fit_input)
```

---

# plot_peak_df *Plot Peaks from a Peak Table*

---

## Description

Plot fits for series of spectra with parameters from a peak data frame

## Usage

```
plot_peak_df(
  peak_df,
  spectra,
  noise_sigma = NULL,
  noise_cutoff = 4,
  omega0_plus = c(0.075, 0.75) * 2,
  cex = 0.2,
  lwd = 0.25,
  label = TRUE,
  label_col = "black",
  p0 = NULL,
  p1 = NULL,
  add = FALSE
)
```

## Arguments

| | |
|---|---|
| peak_df | data frame with peak data as produced by param_list_to_peak_df. |
| spectra | list of spectra corresponding to the volumes found in peak_df. |
| noise_sigma | numeric vector of noise levels associated with each spectrum. If NULL, it is calculated with noise_estimate. |
| noise_cutoff | numeric value used to calculate the lowest contour level according to noise_sigma*noise_cutoff. |
| omega0_plus | omega0 bounds for plotting |
| cex | numeric value by which to scale blue points and labels. |
| lwd | numeric value giving width of contour lines. |
| label | logical indicating whether to draw text labels and connecting lines. |
| label_col | color for text labels |
| p0 | zero order phase for plotting modeled peaks. |
| p1 | first order phase for plotting modeled peaks. |
| add | logical indicating whether to suppress generation of a new plot and add to an existing plot. |

## Details

The raw spectral data is shown in black contours and the modeled peak intensity is shown in red. The centers of peaks are shown with semi-transparent blue dots, with the area of the dot proportional to the volume of the peak (m0). Blue lines connect peaks from modeled doublets. Singlets or doublets are labeled with the syntax <peak>:<fit>. If an F-test p-value column is present (f_pvalue), that will be given below the peak label.

## Value

No return value, called for side effects (draws plots).

## Examples

```
spec_file <- system.file("extdata", "t1", "1.ft2", package = "fitnmr")
spec <- read_nmrpipe(spec_file, dim_order = "hx")

peak_df <- data.frame(
  peak = 1:8,
  fit = c(1, 1, 2, 2, 3, 3, 3, 3),
  f_pvalue = c(4.5664e-10, 1.1190e-05, 1.8765e-15, 5.8171e-04,
               7.5270e-13, 2.5923e-27, 1.3424e-05, 1.4680e-13),
  omega0_ppm_1 = c(8.2476, 8.2596, 8.5400, 8.5202, 8.6124, 8.5853, 8.6449, 8.5370),
  omega0_ppm_2 = c(121.87, 121.93, 119.76, 119.73, 123.36, 123.03, 123.47, 122.96),
  sc_hz_1 = c(3.2806, 3.2806, 2.0000, 2.0000, 7.6481, 7.6481, 7.6481, 7.6481),
  r2_hz_1 = c(2.9072, 2.9072, 4.7886, 4.7886, 5.2849, 5.2849, 5.2849, 5.2849),
  r2_hz_2 = c(2.3345, 2.3345, 2.0996, 2.0996, 2.1801, 2.1801, 2.1801, 2.1801),
  `1.ft2` = c(824420657, 240560662, 1020008726, 89977216,
              848579189, 607904936, 147984411, 161971930),
  check.names = FALSE
)
```

```
plot_peak_df(peak_df, list("1.ft2" = spec), cex = 0.6)
```

---

`plot_resonances_1d` *Plot resonances from 1D fit*

---

## Description

Plot resonances from 1D fit

## Usage

```
plot_resonances_1d(fit_data, always_show_start = FALSE, omega0_plus = 0.05)
```

## Arguments

| | |
|---|---|
| `fit_data` | fit_input or fit_output structure |
| `always_show_start` | show start parameters even if fit already done |
| `omega0_plus` | length 3 vector giving ppm range for each dimension |

## Value

No return value, called for side effects (draws one or more plots).

## Examples

```
spec_file <- system.file("extdata", "tyrosine", "proton.ft1", package = "fitnmr")
spec <- read_nmrpipe(spec_file)

start_resonances <- structure(
  list(
    x = c("HA", "HB3", "HB2", "HD1/2", "HE1/2"),
    x_sc = c(
      "HA-HB3 HA-HB2",
      "HA-HB3 HB3-HB2",
      "HA-HB2 HB3-HB2",
      "HD1/2-HE1/2-3 HD1/2-HE1/2-5",
      "HD1/2-HE1/2-3 HD1/2-HE1/2-5"
    )
  ),
  row.names = c("HA", "HB3", "HB2", "HD1/2", "HE1/2"),
  class = "data.frame"
)

start_nuclei <- structure(
  list(
    omega0_ppm = c(3.364, 2.635, 2.799, 6.94, 6.538),
```

```
    r2_hz = c(0.7, 0.7, 0.7, 0.7, 0.7)
  ),
  class = "data.frame",
  row.names = c("HA", "HB3", "HB2", "HD1/2", "HE1/2")
)

start_couplings <- structure(
  list(hz = c(7.153, 5.159, -13.941, 7.7, 2)),
  class = "data.frame",
  row.names = c("HA-HB3", "HA-HB2", "HB3-HB2", "HD1/2-HE1/2-3", "HD1/2-HE1/2-5")
)

start_tables <- list(
  resonances = start_resonances,
  nuclei = start_nuclei,
  couplings = start_couplings
)
param_list <- tables_to_param_list(list(spec), start_tables)
param_list$start_list$m0[] <- 1e9
arg_list <- param_list_to_arg_list(param_list)
fit_input <- do.call(make_fit_input, c(list(list(spec), omega0_plus = 0.075), arg_list))
fit_output <- perform_fit(fit_input)
plot_resonances_1d(fit_output, always_show_start = FALSE, omega0_plus = 0.075)
```

---

`plot_resonances_2d` *Plot resonances from 2D fit*

---

## Description

Plot resonances from 2D fit

## Usage

```
plot_resonances_2d(
  fit_data,
  omega0_plus,
  resonances = unique(fit_data$resonance_names),
  low_frac = 0.05,
  field = TRUE,
  proj_frac = 0.2
)
```

## Arguments

| | |
|---|---|
| `fit_data` | fit_input or fit_output structure |
| `omega0_plus` | length 2 vector giving ppm range for each dimension |
| `resonances` | character vector with resonances to plot |

| | |
|---|---|
| low_frac | minimum absolute value (as a fraction of maximum intensity) at which to show contours |
| field | logical indicating whether to show modeling of field inhomogeneity as separate peaks |
| proj_frac | fraction of plot area reserved for 1D projections |

## Value

No return value, called for side effects (draws one plot layout per resonance).

---

# plot_resonances_3d *Plot resonances from 3D fit*

---

## Description

Plot resonances from 3D fit

## Usage

```
plot_resonances_3d(
  fit_data,
  omega0_plus,
  resonances = unique(fit_data$resonance_names)
)
```

## Arguments

| | |
|---|---|
| fit_data | fit_input or fit_output structure |
| omega0_plus | length 3 vector giving ppm range for each dimension |
| resonances | character vector with resonances to plot |

## Value

No return value, called for side effects (draws one set of projections per resonance).

plot_sparse_1d *Plot spectrum from 1D fit*

## Description

Plot spectrum from 1D fit

## Usage

```
plot_sparse_1d(
  fit_data,
  tables = NULL,
  spec_idx = 1,
  col_model = 2,
  col_resonance = NULL,
  lwd = 1,
  tick_spacing = 0.02,
  coupling_spacing = 0.01,
  coupling_marks = 0.009,
  xaxs = "i",
  yaxt = "n",
  bty = "n",
  always_show_start = FALSE,
  add = FALSE,
  ppm_map = make_map(get_spec_int(fit_data, "input", spec_idx)[[1]])
)
```

## Arguments

| | |
|---|---|
| fit_data | fit_input or fit_output structure |
| tables | list with resonances, nuclei, and couplings tables |
| spec_idx | index of spectrum to plot |
| col_model | color for modeled peak shapes |
| col_resonance | colors for resonances |
| lwd | line width |
| tick_spacing | spacing between axis ticks (ppm) |
| coupling_spacing | spacing between coupling annotations (fraction of plot) |
| coupling_marks | size of coupling tick marks (fraction of plot) |
| xaxs | x-axis style passed to plot |
| yaxt | y-axis style passed to plot |
| bty | box type passed to plot |
| always_show_start | logical indicating whether to show starting model when fit is present |

| | |
|---|---|
| add | logical indicating whether to add to existing plot |
| ppm_map | sparse axis map created by make_map |

## Value

No return value, called for side effects (draws a plot).

## Examples

```
spec_file <- system.file("extdata", "tyrosine", "proton.ft1", package = "fitnmr")
spec <- read_nmrpipe(spec_file)

start_resonances <- structure(
  list(
    x = c("HA", "HB3", "HB2", "HD1/2", "HE1/2"),
    x_sc = c(
      "HA-HB3 HA-HB2",
      "HA-HB3 HB3-HB2",
      "HA-HB2 HB3-HB2",
      "HD1/2-HE1/2-3 HD1/2-HE1/2-5",
      "HD1/2-HE1/2-3 HD1/2-HE1/2-5"
    )
  ),
  row.names = c("HA", "HB3", "HB2", "HD1/2", "HE1/2"),
  class = "data.frame"
)

start_nuclei <- structure(
  list(
    omega0_ppm = c(3.364, 2.635, 2.799, 6.94, 6.538),
    r2_hz = c(0.7, 0.7, 0.7, 0.7, 0.7)
  ),
  class = "data.frame",
  row.names = c("HA", "HB3", "HB2", "HD1/2", "HE1/2")
)

start_couplings <- structure(
  list(hz = c(7.153, 5.159, -13.941, 7.7, 2)),
  class = "data.frame",
  row.names = c("HA-HB3", "HA-HB2", "HB3-HB2", "HD1/2-HE1/2-3", "HD1/2-HE1/2-5")
)

start_tables <- list(
  resonances = start_resonances,
  nuclei = start_nuclei,
  couplings = start_couplings
)
param_list <- tables_to_param_list(list(spec), start_tables)
param_list$start_list$m0[] <- 1e9
arg_list <- param_list_to_arg_list(param_list)
fit_input <- do.call(make_fit_input, c(list(list(spec), omega0_plus = 0.075), arg_list))
fit_output <- perform_fit(fit_input)
```

```
plot_sparse_1d(fit_output, start_tables)
```

---

`plot_sparse_2d` *Plot spectrum from 2D fit*

---

## Description

Plot spectrum from 2D fit

## Usage

```
plot_sparse_2d(
  fit_data,
  tables = NULL,
  spec_idx = 1,
  spec_int = NULL,
  col_model = 2,
  col_nucleus = NULL,
  lwd = 1,
  tick_spacing = 0.02,
  coupling_spacing = 0.01,
  coupling_marks = 0.009,
  xaxs = "i",
  yaxs = "i",
  low_frac = 0.05,
  bty = "n",
  always_show_start = FALSE,
  add = FALSE,
  ppm_map_list = NULL
)
```

## Arguments

| | |
|---|---|
| `fit_data` | fit_input or fit_output structure |
| `tables` | list with resonances, nuclei, and couplings tables |
| `spec_idx` | index of spectrum to plot |
| `spec_int` | optional matrix to replace the input spectrum intensities |
| `col_model` | color for modeled peak shapes |
| `col_nucleus` | colors for nuclei |
| `lwd` | line width |
| `tick_spacing` | spacing between axis ticks (ppm) |
| `coupling_spacing` | spacing between coupling annotations (fraction of plot) |
| `coupling_marks` | size of coupling tick marks (fraction of plot) |

| | |
|---|---|
| xaxs | x-axis style passed to plot |
| yaxs | y-axis style passed to plot |
| low_frac | minimum absolute value (as a fraction of maximum intensity) at which to show contours |
| bty | box type passed to plot |
| always_show_start | logical indicating whether to show starting model when fit is present |
| add | logical indicating whether to add to existing plot |
| ppm_map_list | optional list of sparse axis maps created by make_map |

## Value

No return value, called for side effects (draws a plot).

---

ppm_to_pts *Convert PPM values to points*

---

## Description

Convert PPM values to points

## Usage

```
ppm_to_pts(ppm_mat, fheader)
```

## Arguments

| | |
|---|---|
| ppm_mat | matrix of ppm values |
| fheader | matrix of generalized ND parameters |

## Value

A numeric matrix of point indices with the same dimensions as ‘ppm_mat‘, with column names converted from ‘*_PPM‘ to ‘*_AXIS‘.

## read_nmrdraw_peak_tab *Read an NMRDraw formatted peak table*

### Description

Read an NMRDraw formatted peak table

### Usage

```
read_nmrdraw_peak_tab(file_path)
```

### Arguments

| | |
|---|---|
| file_path | path to the NMRDraw peak table file |

### Value

A ‘data.frame‘ containing the peak table, with columns defined by the ‘VARS‘ line in the NMRDraw file.

## read_nmrpipe *Read NMRPipe spectrum*

### Description

This function reads 1D-4D spectra stored in the NMRPipe format.

### Usage

```
read_nmrpipe(inFormat, dim_order = NULL, complex_data = FALSE)
```

### Arguments

| | |
|---|---|
| inFormat | character with file name or format for multiple files |
| dim_order | integer vector used to reorder dimensions or character specifying |
| complex_data | logical value indicating whether complex data should be read |

## Details

For three and four dimensional datasets, the spectral data is often spread across multiple files. To read those, inFormat should be a sprintf-style string that describes how the files are named. For instance, if the files are named 001.ft3, 002.ft3, etc., then `inFormat` should be `"%03i.ft3"`. If there is no zero-padding, as in this case, 0 should be omitted from the format. If there are fewer digits, then the first 3 should be changed accordingly.

The default 2D NMRPipe scripts only have a single transpose ("TP") command, leaving the the indirect dimension as the first dimension in the resulting array. The 2D plotting functions in `fitnmr` usually plot this first dimension along the x-axis, which will make for generally non-standard contour plots. Furthermore, when peak fitting is employed, this will also be the first dimension. To fix this, you can change the spectral order with the `dim_order` parameter. The order of the dimensions should be specified in the same way that would be done for aperm. Alternatively, you can specify a character argument to have `fitnmr` attempt to automatically detect and correct the array order. The only currently supported type is `"hx"`, which will put the dimension with the greatest observe frequency first.

This function is partly based on the pipe2rnmr function from rNMR.

## Value

a named list with four elements:

**int** multidimensional array with spectrum intensities (ppm values are given in the dimnames)

**ppm** list of numeric vectors giving the ppm values associated with each int array dimension

**fheader** matrix with dimension-specific header information

**header** numeric vector with the complete header contents

The `fheader` row definitions are as follows (taken from NMRPipe fdatp.h):

| Row Name | Description |
|---|---|
| SIZE | Number of points in dimension |
| APOD | Current valid time-domain size |
| SW | Sweep Width, Hz |
| ORIG | Axis Origin (Last Point), Hz |
| OBS | Obs Freq, MHz |
| FTFLAG | 1=Freq Domain 0=Time Domain |
| QUADFLAG | Data Type Code (See Below) |
| UNITS | Axis Units Code (See Below) |
| P0 | Zero Order Phase, Degrees |
| P1 | First Order Phase, Degrees |
| CAR | Carrier Position, PPM |
| CENTER | Point Location of Zero Freq |
| AQSIGN | Sign adjustment needed for FT |
| APODCODE | Window function used |
| APODQ1 | Window parameter 1 |
| APODQ2 | Window parameter 2 |
| APODQ3 | Window parameter 3 |
| C1 | Add 1.0 to get First Point Scale |

| | |
|---|---|
| ZF | Negative of Zero Fill Size |
| X1 | Extract region origin, if any, pts |
| XN | Extract region endpoint, if any, pts |
| OFFPPM | Additional PPM offset (for alignment) |
| FTSIZE | Size of data when FT performed |
| TDSIZE | Original valid time-domain size |
| LB | Extra Exponential Broadening, Hz |
| GB | Extra Gaussian Broadening, Hz |
| GOFF | Offset for Gaussian Broadening, 0 to 1 |
| OBSMID | Original Obs Freq before 0.0ppm adjust |

In addition several rows contain information inferred from the header data:

| Row Name | Description |
|---|---|
| aq_s | Acquisition time, seconds |
| sw_ppm | Original Sweep Width, PPM |
| direct | Direct (1) or indirect (0) dimension |
| alias | Aliasing (0/1) with inversion (-1) |
| mag | Magnitude mode (direct from FDMCFLAG) |

## Examples

```
spec_file <- system.file("extdata", "t1", "1.ft2", package="fitnmr")
spec <- read_nmrpipe(spec_file, dim_order="hx")
str(spec)
```

---

# read_resonance_tables *Read resonance, nuclei, coupling, and spin system tables*

---

## Description

Read resonance, nuclei, coupling, and spin system tables

## Usage

```
read_resonance_tables(
  resonances_file = "start_resonances.csv",
  nuclei_file = "start_nuclei.csv",
  couplings_file = "start_couplings.csv",
  comment.char = ""
)
```

### Arguments

| | |
|---|---|
| resonances_file | path to resonances CSV file |
| nuclei_file | path to nuclei CSV file |
| couplings_file | path to couplings CSV file |
| comment.char | comment character passed to read.csv |

---

## resonance_to_param_list *Convert data frame of resonances into a parameter list*

---

### Description

Convert data frame of resonances into a parameter list

### Usage

```
resonance_to_param_list(spec, resonance, nuclei, couplings)
```

### Arguments

| | |
|---|---|
| spec | single spectrum |
| resonance | data frame with resonances |
| nuclei | data frame with nuclei chemical shifts and R2 rates |
| couplings | data frame with scalar couplings |

---

## set_spinsystem_params *Update HamiltonianMultiplet parameters from input data*

---

### Description

Update HamiltonianMultiplet parameters from input data

### Usage

```
set_spinsystem_params(
  spinsystems,
  omega0,
  omega0_comb,
  coupling_values = NULL,
  nucleus_names,
  ref_freq,
  spec_idx = 1
)
```

### Arguments

| | |
|---|---|
| spinsystems | named list of HamiltonianMultiplet objects |
| omega0 | array of chemical shifts (ppm) |
| omega0_comb | named vector of coupling values (Hz) |
| coupling_values | named vector of coupling values (Hz) |
| nucleus_names | array of nucleus names aligned with omega0 |
| ref_freq | numeric vector of reference frequencies (MHz) |
| spec_idx | spectrum index to use for chemical shift conversions |

---

## sim_time_nd *Simulate an FID using the NMRPipe SimTimeND function*

---

### Description

Simulate an FID using the NMRPipe SimTimeND function

### Usage

```
sim_time_nd(
  peak_tab,
  fheader,
  rms = 0,
  iseed = stats::runif(1, max = .Machine$integer.max),
  file_path = NULL,
  verbose = FALSE
)
```

### Arguments

| | |
|---|---|
| peak_tab | NMRDraw peak table |
| fheader | matrix of generalized ND parameters |
| rms | RMS noise level |
| iseed | random seed |
| file_path | optional output path for the simulated FID |
| verbose | logical indicating whether to print the command |

### Value

The integer exit status from ‘system2("SimTimeND", ...)‘ (typically ‘0‘ on success).

## spec_bind *Combine parameter lists referring to different spectra*

### Description

Combine parameter lists referring to different spectra

### Usage

```
spec_bind(...)
```

### Arguments

| | |
|---|---|
| ... | any number of parameter lists or a single list of parameter lists |

## spec_overlap_mat *Determine a matrix of fractional peak overlap*

### Description

Determine a matrix of fractional peak overlap

### Usage

```
spec_overlap_mat(peak_int_list)
```

### Arguments

| | |
|---|---|
| peak_int_list | list of peak intensity arrays |

## split_coupling_names *Split string of scalar coupling names*

### Description

The only currently implemented way of splitting string is using whitespace

### Usage

```
split_coupling_names(name_char)
```

### Arguments

| | |
|---|---|
| name_char | single string (character vector of length 1) with scalar coupling names |

tables_to_param_list *Convert tables with resonance/nuclei/couplings to a parameter list*

## Description

Convert tables with resonance/nuclei/couplings to a parameter list

## Usage

```
tables_to_param_list(spec_list, tables)
```

## Arguments

| | |
|---|---|
| spec_list | list of spectra |
| tables | list with resonance/nuclei/couplings tables |

## Value

A parameter 'list' containing 'start_list', 'group_list', 'comb_list', and associated naming vectors used by fitting functions.

## Examples

```
spec_file <- system.file("extdata", "tyrosine", "proton.ft1", package = "fitnmr")
spec <- read_nmrpipe(spec_file)

start_resonances <- structure(
  list(
    x = c("HA", "HB3", "HB2", "HD1/2", "HE1/2"),
    x_sc = c(
      "HA-HB3 HA-HB2",
      "HA-HB3 HB3-HB2",
      "HA-HB2 HB3-HB2",
      "HD1/2-HE1/2-3 HD1/2-HE1/2-5",
      "HD1/2-HE1/2-3 HD1/2-HE1/2-5"
    )
  ),
  row.names = c("HA", "HB3", "HB2", "HD1/2", "HE1/2"),
  class = "data.frame"
)

start_nuclei <- structure(
  list(
    omega0_ppm = c(3.364, 2.635, 2.799, 6.94, 6.538),
    r2_hz = c(0.7, 0.7, 0.7, 0.7, 0.7)
  ),
  class = "data.frame",
  row.names = c("HA", "HB3", "HB2", "HD1/2", "HE1/2")
)
```

```
start_couplings <- structure(
  list(hz = c(7.153, 5.159, -13.941, 7.7, 2)),
  class = "data.frame",
  row.names = c("HA-HB3", "HA-HB2", "HB3-HB2", "HD1/2-HE1/2-3", "HD1/2-HE1/2-5")
)

start_tables <- list(
  resonances = start_resonances,
  nuclei = start_nuclei,
  couplings = start_couplings
)
param_list <- tables_to_param_list(list(spec), start_tables)
param_list$start_list$m0[] <- 1e9
arg_list <- param_list_to_arg_list(param_list)
fit_input <- do.call(make_fit_input, c(list(list(spec), omega0_plus = 0.075), arg_list))
fit_output <- perform_fit(fit_input)
```

---

`update_fit_bounds` *Update bounds on fitting parameters*

---

## Description

Update bounds on fitting parameters

## Usage

```
update_fit_bounds(
  fit_input,
  omega0_r2_factor = NULL,
  r2_bounds = NULL,
  sc_bounds = NULL
)
```

## Arguments

| | |
|---|---|
| `fit_input` | fit_input structure |
| `omega0_r2_factor` | optional factor for omega0 bounds based on r2 |
| `r2_bounds` | optional numeric vector of length 2 |
| `sc_bounds` | optional numeric vector of length 2 |

## Value

A modified ‘fit_input‘ list with updated parameter bounds.

## Examples

```
spec_file <- system.file("extdata", "t1", "1.ft2", package = "fitnmr")
spec <- read_nmrpipe(spec_file, dim_order = "hx")
fit_input <- make_fit_input(
  list(spec),
  omega0_start = matrix(c(8.5400, 119.76), nrow = 2),
  omega0_plus = c(0.075, 0.75),
  r2_start = 4,
  m0_start = 1e9
)
fit_input <- update_fit_bounds(fit_input, omega0_r2_factor=1.5, r2_bounds=c(0.5, 20))
```

---

# update_spinsystem_params *Update HamiltonianMultiplet parameters from fit data*

---

## Description

Update HamiltonianMultiplet parameters from fit data

## Usage

```
update_spinsystem_params(fit_data, spec_idx = 1, param_list = NULL)
```

## Arguments

| | |
|---|---|
| fit_data | fit_input or fit_output structure |
| spec_idx | spectrum index to use for chemical shift conversions |
| param_list | optional parameter list (start_list or fit_list) |

---

# whz_to_pts *Convert widths in Hz into points*

---

## Description

Convert widths in Hz into points

## Usage

```
whz_to_pts(whz_mat, fheader)
```

## Arguments

| | |
|---|---|
| whz_mat | matrix of widths in Hz |
| fheader | matrix of generalized ND parameters |

`write_nmrdraw_peak_tab` *Write an NMRDraw formatted peak table*

## Description

Write an NMRDraw formatted peak table

## Usage

```
write_nmrdraw_peak_tab(peak_tab, file_path)
```

## Arguments

| | |
|---|---|
| `peak_tab` | data frame containing peak table data |
| `file_path` | path to write the NMRDraw peak table file |

## Value

No return value, called for side effects (writes ‘file_path‘).

# Index