Package: AlpsNMR (via r-universe)

June 15, 2024

Type Package

Title Automated spectraL Processing System for NMR

Version 4.7.1

Date 2024-06-02

Encoding UTF-8

Description Reads Bruker NMR data directories both zipped and unzipped. It provides automated and efficient signal processing for untargeted NMR metabolomics. It is able to interpolate the samples, detect outliers, exclude regions, normalize, detect peaks, align the spectra, integrate peaks, manage metadata and visualize the spectra. After spectra proccessing, it can apply multivariate analysis on extracted data. Efficient plotting with 1-D data is also available. Basic reading of 1D ACD/Labs exported JDX samples is also available.

License MIT + file LICENSE

URL https://sipss.github.io/AlpsNMR/, https://github.com/sipss/AlpsNMR

BugReports https://github.com/sipss/AlpsNMR/issues

LazyData FALSE

Depends R (>= 4.2)

Imports utils, generics, graphics, stats, grDevices, cli, magrittr (>= 1.5), dplyr (>= 1.1.0), signal (>= 0.7-6), rlang (>= 0.3.0.1), scales (>= 1.2.0), stringr (>= 1.3.1), tibble(>= 1.3.4), tidyr (>= 1.0.0), tidyselect, readxl (>= 1.1.0), purrr (>= 0.2.5), glue (>= 1.2.0), reshape2 (>= 1.4.3), mixOmics (>= 6.22.0), matrixStats (>= 0.54.0), fs (>= 1.2.6), rmarkdown (>= 1.10), speaq (>= 2.4.0), htmltools (>= 0.3.6), pcaPP (>= 1.9-73), ggplot2 (>= 3.1.0), baseline (>= 1.2-1), vctrs (>= 0.3.0), BiocParallel (>= 1.34.0)

Suggests ASICS, BiocStyle, ChemoSpec, cowplot, curl, DT (>= 0.5), GGally (>= 1.4.0), ggrepel (>= 0.8.0), gridExtra, knitr, NMRphasing, plotly (>= 4.7.1), progressr, SummarizedExperiment, S4Vectors, testthat (>= 2.0.0), writexl (>= 1.0), zip (>= 2.0.4)

biocViews Software, Preprocessing, Visualization, Classification, Cheminformatics, Metabolomics, DataImport

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Repository https://zeehio.r-universe.dev

RemoteUrl https://github.com/sipss/AlpsNMR

RemoteRef HEAD

RemoteSha 9a68fc7858122dc478239d4c622165498d26b42a

Contents

`AlpsNMR-package` *AlpsNMR: Automated spectraL Processing System for NMR*

Description

AlpsNMR allows you to import NMR spectra into R and provides automated and efficient signal processing for untargeted NMR metabolomics.

Details

The following functions can be combined with the pipe. They create or modify the nmr_dataset object.

- `nmr_read_samples_dir()` or `nmr_read_samples()`
- `nmr_interpolate_1D()`
- `nmr_exclude_region()`
- `nmr_normalize()`
- plot()

There are also functions to extract the metadata and submit the samples to irods, see the example below.

The nmr_dataset object is essentially a list, so it is easy to access its components for further analysis.

Author(s)

Maintainer: Sergio Oller Moreno <sergioller@gmail.com> (ORCID)

Authors:

- Ivan Montoliu Roura <Ivan.MontoliuRoura@rd.nestle.com>
- Francisco Madrid Gambin <fmadrid@ibecbarcelona.eu> (ORCID)
- Luis Fernandez <lfernandez@ibecbarcelona.eu> (ORCID)
- Héctor Gracia Cabrera <hgracia@ibecbarcelona.eu>
- Santiago Marco Colás <smarco@ibecbarcelona.eu> (ORCID)

Other contributors:

- Laura López Sánchez [contributor]
- Nestlé Institute of Health Sciences [copyright holder]
- Institute for Bioengineering of Catalonia [copyright holder]
- Miller Jack <jack.miller@physics.org> (ORCID) (Autophase wrapper, ASICS export) [contributor]

See Also

Useful links:

- https://sipss.github.io/AlpsNMR/
- https://github.com/sipss/AlpsNMR
- Report bugs at https://github.com/sipss/AlpsNMR/issues

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
my_nmr_dataset <- dataset %>%
    nmr_interpolate_1D(axis = c(0.4, 10)) %>%
    nmr_exclude_region(exclude = list(water = c(4.6, 5))) %>%
    nmr_normalize(method = "pqn") %>%
    plot()
```

---

bp_kfold_VIP_analysis *K-fold bootstrap and permutation over PLS-VIP*

Description

Bootstrap and permutation over PLS-VIP on AlpsNMR can be performed on both nmr_dataset_1D full spectra as well as nmr_dataset_peak_table peak tables.

Usage

```
bp_kfold_VIP_analysis(dataset, y_column, k = 4, ncomp = 3, nbootstrap = 300)
```

Arguments

dataset	An nmr_dataset_family object
y_column	A string with the name of the y column (present in the metadata of the dataset)
k	Number of folds, recomended between 4 to 10
ncomp	number of components for the bootstrap models
nbootstrap	number of bootstrap dataset

Details

Use of the bootstrap and permutation methods for a more robust variable importance in the projection metric for partial least squares regression, in a k-fold cross validation

Value

A list with the following elements:

- important_vips: A list with the important vips selected
- relevant_vips: List of vips with some relevance
- wilcoxon_vips: List of vips that pass a wilcoxon test
- vip_means: Means of the vips scores
- vip_score_plot: plot of the vips scores
- kfold_resuls: results of the k bp_VIP_analysis
- kfold_index: list of index of partitions of the folds

Examples

```
# Data analysis for a table of integrated peaks
set.seed(42)
## Generate an artificial nmr_dataset_peak_table:
### Generate artificial metadata:
num_samples <- 64 # use an even number in this example
num_peaks <- 10
metadata <- data.frame(
    NMRExperiment = as.character(1:num_samples),
    Condition = sample(rep(c("A", "B"), times = num_samples / 2), num_samples)
)

### The matrix with peaks
peak_means <- runif(n = num_peaks, min = 300, max = 600)
peak_sd <- runif(n = num_peaks, min = 30, max = 60)
peak_matrix <- mapply(function(mu, sd) rnorm(num_samples, mu, sd),
    mu = peak_means, sd = peak_sd
)
colnames(peak_matrix) <- paste0("Peak", 1:num_peaks)
rownames(peak_matrix) <- paste0("Sample", 1:num_samples)

## Artificial differences depending on the condition:
peak_matrix[metadata$Condition == "A", "Peak2"] <-
    peak_matrix[metadata$Condition == "A", "Peak2"] + 70

peak_matrix[metadata$Condition == "A", "Peak6"] <-
    peak_matrix[metadata$Condition == "A", "Peak6"] - 60

### The nmr_dataset_peak_table
peak_table <- new_nmr_dataset_peak_table(
    peak_table = peak_matrix,
    metadata = list(external = metadata)
)

## We will use bootstrap and permutation method for VIPs selection
## in a a k-fold cross validation
bp_results <- bp_kfold_VIP_analysis(peak_table, # Data to be analyzed
    y_column = "Condition", # Label
    k = 2,
    ncomp = 1,
    nbootstrap = 5
)

message("Selected VIPs are: ", bp_results$important_vips)
```

---

bp_VIP_analysis *Bootstrap and permutation over PLS-VIP*

Description

Bootstrap and permutation over PLS-VIP on AlpsNMR can be performed on both nmr_dataset_1D full spectra as well as nmr_dataset_peak_table peak tables.

Usage

```
bp_VIP_analysis(dataset, train_index, y_column, ncomp, nbootstrap = 300)
```

Arguments

dataset	An nmr_dataset_family object
train_index	set of index used to generate the bootstrap datasets
y_column	A string with the name of the y column (present in the metadata of the dataset)
ncomp	number of components used in the plsda models
nbootstrap	number of bootstrap dataset

Details

Use of the bootstrap and permutation methods for a more robust variable importance in the projection metric for partial least squares regression

Value

A list with the following elements:

- important_vips: A list with the important vips selected
- relevant_vips: List of vips with some relevance
- pls_vip: Pls-VIPs of every bootstrap
- pls_vip_perm: Pls-VIPs of every bootstrap with permuted variables
- pls_vip_means: Pls-VIPs normaliced differences means
- pls_vip_score_diff: Differences of pls_vip and pls_vip_perm
- pls_models: pls models of the diferent bootstraps
- pls_perm_models: pls permuted models of the diferent bootstraps
- classif_rate: classification rate of the bootstrap models
- general_model: pls model trained with all train data
- general_CR: classification rate of the general_model
- vips_model: pls model trained with vips selection over all train data
- vips_CR: classification rate of the vips_model
- error: error spected in a t distribution
- lower_bound: lower bound of the confidence interval
- upper_bound: upper bound of the confidence interval

Examples

```
# Data analysis for a table of integrated peaks

## Generate an artificial nmr_dataset_peak_table:
### Generate artificial metadata:
num_samples <- 32 # use an even number in this example
num_peaks <- 20
metadata <- data.frame(
    NMRExperiment = as.character(1:num_samples),
    Condition = rep(c("A", "B"), times = num_samples / 2)
)

### The matrix with peaks
peak_means <- runif(n = num_peaks, min = 300, max = 600)
peak_sd <- runif(n = num_peaks, min = 30, max = 60)
peak_matrix <- mapply(function(mu, sd) rnorm(num_samples, mu, sd),
    mu = peak_means, sd = peak_sd
)
colnames(peak_matrix) <- paste0("Peak", 1:num_peaks)

## Artificial differences depending on the condition:
peak_matrix[metadata$Condition == "A", "Peak2"] <-
    peak_matrix[metadata$Condition == "A", "Peak2"] + 70

peak_matrix[metadata$Condition == "A", "Peak6"] <-
    peak_matrix[metadata$Condition == "A", "Peak6"] - 60

### The nmr_dataset_peak_table
peak_table <- new_nmr_dataset_peak_table(
    peak_table = peak_matrix,
    metadata = list(external = metadata)
)

## We will use a double cross validation, splitting the samples with random
## subsampling both in the external and internal validation.
## The classification model will be a PLSDA, exploring at maximum 3 latent
## variables.
## The best model will be selected based on the area under the ROC curve
methodology <- plsda_auroc_vip_method(ncomp = 3)
model <- nmr_data_analysis(
    peak_table,
    y_column = "Condition",
    identity_column = NULL,
    external_val = list(iterations = 1, test_size = 0.25),
    internal_val = list(iterations = 3, test_size = 0.25),
    data_analysis_method = methodology
)
## Area under ROC for each outer cross-validation iteration:
model$outer_cv_results_digested$auroc

## The number of components for the bootstrap models is selected
ncomps <- model$outer_cv_results$`1`$model$ncomp
```

```
train_index <- model$train_test_partitions$outer$`1`$outer_train

# Bootstrap and permutation for VIP selection
bp_VIPS <- bp_VIP_analysis(peak_table, # Data to be analyzed
    train_index,
    y_column = "Condition",
    ncomp = ncomps,
    nbootstrap = 10
)
```

---

download_MTBLS242 *Download MTBLS242*

Description

Downloads the MTBLS242 dataset from Gralka et al., 2015. DOI: doi:10.3945/ajcn.115.110536.

Usage

```
download_MTBLS242(
  dest_dir = "MTBLS242",
  force = FALSE,
  keep_only_CPMG_1r = TRUE,
  keep_only_preop_and_3months = TRUE,
  keep_only_complete_time_points = TRUE
)
```

Arguments

dest_dir	Directory where the dataset should be saved
force	Logical. If TRUE we do not re-download files if they exist. The function does not check whether cached versions were downloaded with different keep_only_* arguments, so please use force = TRUE if you change the keep_only_* settings.
keep_only_CPMG_1r	If TRUE, remove all other data beyond the CPMG real spectrum, which is enough for the tutorial
keep_only_preop_and_3months	If TRUE, keep only the preoperatory and the "three months after surgery" time points, enough for the tutorial
keep_only_complete_time_points	If TRUE, remove samples that do not appear on all timepoints. Useful for the tutorial.

Details

Besides the destination directory, this function includes three logical parameters to limit the amount of downloaded/saved data. To run the tutorial workflow:

- only the "preop" and "three months" timepoints are used,
- only subjects measured in *both* preop and three months time points are used
- only the CPMG samples are used.

If you want to run the tutorial, you can set those filters to `TRUE`. Then, roughly 800MB will be downloaded, and 77MB of disk space will be used, since for each downloaded sample we remove all the data but the CPMG.

If you set those filters to `FALSE`, roughly 1.8GB of data will be downloaded (since we have more timepoints to download) and 1.8GB of disk space will be used.

Note that we have experienced some sporadic timeouts from Metabolights, when downloading the dataset. If you get those timeouts simply re-run the download function and it will restart from where it stopped.

Note as well, that we observed several files to have incorrect data:

- Obs4_0346s.zip is not present in the FTP server
- Obs0_0110s.zip and Obs1_0256s.zip incorrectly contain sample Obs1_0010s

This function removes all three samples from the samples annotations and doesn't download their data.

Value

Invisibly, the annotations. See the example for how to download the annotations and create a dataset from the downloaded files.

Examples

```
## Not run:
  download_MTBLS242("./MTBLS242")
  annot <- readr::read_tsv(annotations_destfile)

  dataset <- nmr_read_samples(annot$filename)
  dataset <- nmr_meta_add(dataset, annot)
  dataset

## End(Not run)
```

`files_to_rDolphin` *Files to rDoplhin*

Description

The rDolphin family functions are introduced to perform automatic targeted metabolite profiling. Therefore, ensure that you interpolated from -0.1 ppm in order to consider the TSP/DSS signal at 0.0 ppm. The function generates a list with the files required by to_rDolphin function. Then, it is required to save them with the `save_files_to_rDolphin`. to_rDolphin function will read the generated "parameters.csv" file. function.

Usage

```
files_to_rDolphin(nmr_dataset, biological_origin)
```

Arguments

`nmr_dataset`	An nmr_dataset object
`biological_origin`	String specify the type of sample (blood, urine, cell)

Value

a list containing:

- `meta_rDolphin`: metadata in rDolphin format,
- `NMR_spectra`: spectra matrix
- `ROI`: ROI template
- `Parameters`: parameters file

See Also

Other import/export functions: `Pipelines`, `load_and_save_functions`, `nmr_data()`, `nmr_meta_export()`, `nmr_read_bruker_fid()`, `nmr_read_samples()`, `nmr_zip_bruker_samples()`, `save_files_to_rDolphin()`, `save_profiling_output()`, `to_ChemoSpec()`

Examples

```
## Not run:
# Set the directory in which rDolphin files will be saved
output_dir_10_rDolphin <- file.path(your_path, "10-rDolphin")
fs::dir_create(output_dir_10_rDolphin)

# Generate the files (for plasma/serum)
files_rDolphin <- files_to_rDolphin(nmr_dataset_0_10_ppm, blood)

# Save the files
save_files_to_rDolphin(files_rDolphin, output_dir_10_rDolphin)
```

```
# Build the rDolphin object. Do not forget to set the directory
setwd(output_dir_10_rDolphin)
rDolphin_object <- to_rDolphin("Parameters.csv")

# Visualize your spectra
rDolphin_plot(rDolphin_object)

# Run the main profiling function (it takes a while)
targeted_profiling <- Automatic_targeted_profiling(rDolphin_object)

# Save results
save_profiling_output(targeted_profiling, output_dir_10_rDolphin)

save_profiling_plots(
    output_dir_10_rDolphin, targeted_profiling$final_output,
    targeted_profiling$reproducibility_data
)

# Additionally, you can run some stats
intensities <- targeted_profiling$final_output$intensity
group <- as.factor(rDolphin_object$Metadata$type)
model_PLS <- rdCV_PLS_RF(X = intensities, Y = group)

## End(Not run)
```

`file_lister` *NMR file lister*

Description

The function lists samples from the chosen folder required to import and create a nmr_dataset_1D object. The function is based on the `fs::dir_ls()` function.

Usage

```
file_lister(dataset_path_nmr, glob)
```

Arguments

`dataset_path_nmr`	A character vector of the path where samples are.
`glob`	A wildcard or globbing pattern common for the samples to be read, for example ending with *0 (spectra acquired by a NOESY sequence often end by 0: 10, 20, 30...) or *s (for example, samples from the tutorial in this package) passed on to `grep()` to filter paths.

Value

lists of samples from the chosen folder

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
lists_of_samples <- file_lister(dir_to_demo_dataset, "*0")
```

`filter.nmr_dataset_family` *Keep samples based on metadata column criteria*

Description

Keep samples based on metadata column criteria

Usage

```
## S3 method for class 'nmr_dataset_family'
filter(.data, ...)
```

Arguments

`.data`	An nmr_dataset_family object
`...`	conditions, as in dplyr

Value

The same object, with the matching rows

See Also

Other subsetting functions: `[.nmr_dataset()`, `[.nmr_dataset_1D()`, `[.nmr_dataset_peak_table()`, `nmr_pca_outliers_filter()`

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))

## example 1
sample_10 <- filter(dataset_1D, NMRExperiment == "10")

## example 2
# test_samples <- dataset_1D %>% filter(nmr_peak_table$metadata$external$Group == "placebo")
```

format.nmr_dataset *Format for nmr_dataset*

Description

Format for nmr_dataset

Usage

```
## S3 method for class 'nmr_dataset'
format(x, ...)
```

Arguments

x	an nmr_dataset object
...	for future use

Value

Format for nmr_dataset

See Also

Other class helper functions: format.nmr_dataset_1D(), format.nmr_dataset_peak_table(), is.nmr_dataset_1D(), is.nmr_dataset_peak_table(), new_nmr_dataset(), new_nmr_dataset_1D(), new_nmr_dataset_peak_table(), print.nmr_dataset(), print.nmr_dataset_1D(), print.nmr_dataset_peak_table(), validate_nmr_dataset(), validate_nmr_dataset_family(), validate_nmr_dataset_peak_table()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
format(dataset)
```

format.nmr_dataset_1D *format for nmr_dataset_1D*

Description

format for nmr_dataset_1D

Usage

```
## S3 method for class 'nmr_dataset_1D'
format(x, ...)
```

Arguments

x	an nmr_dataset_1D object
...	for future use

Value

format for nmr_dataset_1D

See Also

Other class helper functions: format.nmr_dataset(), format.nmr_dataset_peak_table(), is.nmr_dataset_1D(), is.nmr_dataset_peak_table(), new_nmr_dataset(), new_nmr_dataset_1D(), new_nmr_dataset_peak_table(), print.nmr_dataset(), print.nmr_dataset_1D(), print.nmr_dataset_peak_table(), validate_nmr_dataset(), validate_nmr_dataset_family(), validate_nmr_dataset_peak_table()

Other nmr_dataset_1D functions: [.nmr_dataset_1D(), get_integration_with_metadata(), is.nmr_dataset_1D(), nmr_integrate_peak_positions(), nmr_integrate_regions(), nmr_meta_add(), nmr_meta_export(), nmr_meta_get(), nmr_meta_get_column(), nmr_ppm_resolution(), print.nmr_dataset_1D()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
format(dataset_1D)
```

---

format.nmr_dataset_peak_table

Format for nmr_dataset_peak_table

Description

Format for nmr_dataset_peak_table

Usage

```
## S3 method for class 'nmr_dataset_peak_table'
format(x, ...)
```

Arguments

x	an nmr_dataset_peak_table object
...	for future use

Value

Format for nmr_dataset_peak_table

See Also

Other class helper functions: format.nmr_dataset(), format.nmr_dataset_1D(), is.nmr_dataset_1D(), is.nmr_dataset_peak_table(), new_nmr_dataset(), new_nmr_dataset_1D(), new_nmr_dataset_peak_table(), print.nmr_dataset(), print.nmr_dataset_1D(), print.nmr_dataset_peak_table(), validate_nmr_dataset(), validate_nmr_dataset_family(), validate_nmr_dataset_peak_table()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
meta <- file.path(dir_to_demo_dataset, "dummy_metadata.xlsx")
metadata <- readxl::read_excel(meta, sheet = 1)
dataset_1D <- nmr_meta_add(dataset_1D, metadata = metadata, by = "NMRExperiment")
metadata <- list(external = dataset_1D[["metadata"]][["external"]])
peak_table <- nmr_data(dataset_1D)
new <- new_nmr_dataset_peak_table(peak_table, metadata)
format(new)
```

---

get_integration_with_metadata

Get integrals with metadata from integrate peak positions

Description

Get integrals with metadata from integrate peak positions

Usage

```
get_integration_with_metadata(integration_object)
```

Arguments

integration_object
: A nmr_dataset object

Value

Get integrals with metadata from integrate peak positions

integration dataframe

See Also

Other peak integration functions: Pipelines, nmr_identify_regions_blood(), nmr_identify_regions_cell(), nmr_identify_regions_urine(), nmr_integrate_peak_positions(), nmr_integrate_regions()

Other nmr_dataset_1D functions: [.nmr_dataset_1D(), format.nmr_dataset_1D(), is.nmr_dataset_1D(), nmr_integrate_peak_positions(), nmr_integrate_regions(), nmr_meta_add(), nmr_meta_export(), nmr_meta_get(), nmr_meta_get_column(), nmr_ppm_resolution(), print.nmr_dataset_1D()

Examples

```
peak_table <- matrix(1:6, nrow = 2, ncol = 3)
rownames(peak_table) <- c("10", "20")
colnames(peak_table) <- c("ppm_1.2", "ppm1.4", "ppm1.6")

dataset <- new_nmr_dataset_peak_table(
    peak_table = peak_table,
   metadata = list(external = data.frame(NMRExperiment = c("10", "20"), Condition = c("A", "B")))
)
get_integration_with_metadata(dataset)
```

---

hmdb *The Human Metabolome DataBase multiplet table*

Description

The Human Metabolome DataBase multiplet table

References

https://hmdb.ca/

Examples

```
# Get all the 1-Methylhistidine peaks:
data("hmdb")
hmdb[hmdb$Metabolite == "1-Methylhistidine", ]
```

HMDB_blood *The Human Metabolome DataBase multiplet table: blood metabolites normally found in NMR-based metabolomics*

Description

The Human Metabolome DataBase multiplet table: blood metabolites normally found in NMR-based metabolomics

References

https://hmdb.ca/

Examples

```
data("HMDB_blood")
HMDB_blood[HMDB_blood$Metabolite == "1-Methylhistidine", ]
```

HMDB_cell *The Human Metabolome DataBase multiplet table: cell metabolites normally found in NMR-based metabolomics*

Description

The Human Metabolome DataBase multiplet table: cell metabolites normally found in NMR-based metabolomics

References

https://hmdb.ca/

Examples

```
data("HMDB_cell")
HMDB_cell[HMDB_cell$Metabolite == "Acetone", ]
```

HMDB_urine *The Human Metabolome DataBase multiplet table: urine metabolites normally found in NMR-based metabolomics*

Description

The Human Metabolome DataBase multiplet table: urine metabolites normally found in NMR-based metabolomics

References

https://hmdb.ca/

Examples

```
data("HMDB_urine")
HMDB_urine[HMDB_urine$Metabolite == "1-Methyladenosine", ]
```

is.nmr_dataset *Object is of nmr_dataset class*

Description

Object is of nmr_dataset class

Usage

```
is.nmr_dataset(x)
```

Arguments

x	An object

Value

TRUE if the object is an nmr_dataset, FALSE otherwise

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
is(dataset)
```

is.nmr_dataset_1D *Object is of nmr_dataset_1D class*

Description

Object is of nmr_dataset_1D class

Usage

```
is.nmr_dataset_1D(x)
```

Arguments

x	an nmr_dataset_1D object

Value

TRUE if the object is an nmr_dataset_1D, FALSE otherwise

See Also

Other class helper functions: format.nmr_dataset(), format.nmr_dataset_1D(), format.nmr_dataset_peak_table(), is.nmr_dataset_peak_table(), new_nmr_dataset(), new_nmr_dataset_1D(), new_nmr_dataset_peak_table(), print.nmr_dataset(), print.nmr_dataset_1D(), print.nmr_dataset_peak_table(), validate_nmr_dataset(), validate_nmr_dataset_family(), validate_nmr_dataset_peak_table()

Other nmr_dataset_1D functions: [.nmr_dataset_1D(), format.nmr_dataset_1D(), get_integration_with_metadata(), nmr_integrate_peak_positions(), nmr_integrate_regions(), nmr_meta_add(), nmr_meta_export(), nmr_meta_get(), nmr_meta_get_column(), nmr_ppm_resolution(), print.nmr_dataset_1D()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
result <- is(dataset_1D)
```

---

is.nmr_dataset_peak_table

Object is of nmr_dataset_peak_table class

Description

Object is of nmr_dataset_peak_table class

Usage

```
is.nmr_dataset_peak_table(x)
```

Arguments

x	an nmr_dataset_peak_table object

Value

TRUE if the object is an nmr_dataset_peak_table, FALSE otherwise

See Also

Other class helper functions: format.nmr_dataset(), format.nmr_dataset_1D(), format.nmr_dataset_peak_table(), is.nmr_dataset_1D(), new_nmr_dataset(), new_nmr_dataset_1D(), new_nmr_dataset_peak_table(), print.nmr_dataset(), print.nmr_dataset_1D(), print.nmr_dataset_peak_table(), validate_nmr_dataset(), validate_nmr_dataset_family(), validate_nmr_dataset_peak_table()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
meta <- file.path(dir_to_demo_dataset, "dummy_metadata.xlsx")
metadata <- readxl::read_excel(meta, sheet = 1)
dataset_1D <- nmr_meta_add(dataset_1D, metadata = metadata, by = "NMRExperiment")
metadata <- list(external = dataset_1D[["metadata"]][["external"]])
peak_table <- nmr_data(dataset_1D)
new <- new_nmr_dataset_peak_table(peak_table, metadata)
is(new)
```

`load_and_save_functions`

nmr_dataset_load

Description

nmr_dataset_load

nmr_dataset_save

Usage

```
nmr_dataset_load(file_name)

nmr_dataset_save(nmr_dataset, file_name, ...)
```

Arguments

`file_name`	The file name to load or save to
`nmr_dataset`	An object from the nmr_dataset_family
`...`	Additional arguments passed to saveRDS.

Value

Functions to load and save nmr_dataset objects

load nmr dataset

save nmr dataset

See Also

Other import/export functions: `Pipelines`, `files_to_rDolphin()`, `nmr_data()`, `nmr_meta_export()`, `nmr_read_bruker_fid()`, `nmr_read_samples()`, `nmr_zip_bruker_samples()`, `save_files_to_rDolphin()`, `save_profiling_output()`, `to_ChemoSpec()`

Examples

```
# dataset <- nmr_dataset_load("test")
nmr_dataset <- nmr_dataset_load(system.file("extdata", "nmr_dataset.rds", package = "AlpsNMR"))

dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
# nmr_dataset_save(dataset, "test")
```

`models_stability_plot_bootstrap` *Models stability plot*

Description

Plot stability among models of the external cross validation

Usage

```
models_stability_plot_bootstrap(bp_results)
```

Arguments

`bp_results`	bp_kfold_VIP_analysis results

Value

A plot of models stability

Examples

```
# Data analysis for a table of integrated peaks

## Generate an artificial nmr_dataset_peak_table:
### Generate artificial metadata:
num_samples <- 64 # use an even number in this example
num_peaks <- 20
metadata <- data.frame(
    NMRExperiment = as.character(1:num_samples),
    Condition = rep(c("A", "B"), times = num_samples / 2)
)

### The matrix with peaks
peak_means <- runif(n = num_peaks, min = 300, max = 600)
peak_sd <- runif(n = num_peaks, min = 30, max = 60)
peak_matrix <- mapply(function(mu, sd) rnorm(num_samples, mu, sd),
    mu = peak_means, sd = peak_sd
)
colnames(peak_matrix) <- paste0("Peak", 1:num_peaks)
```

```
## Artificial differences depending on the condition:
peak_matrix[metadata$Condition == "A", "Peak2"] <-
    peak_matrix[metadata$Condition == "A", "Peak2"] + 70

peak_matrix[metadata$Condition == "A", "Peak6"] <-
    peak_matrix[metadata$Condition == "A", "Peak6"] - 60

### The nmr_dataset_peak_table
peak_table <- new_nmr_dataset_peak_table(
    peak_table = peak_matrix,
    metadata = list(external = metadata)
)

## We will use bootstrap and permutation method for VIPs selection
## in a a k-fold cross validation
# bp_results <- bp_kfold_VIP_analysis(peak_table, # Data to be analized
#                             y_column = "Condition", # Label
#                             k = 3,
#                             nbootstrap = 10)

# message("Selected VIPs are: ", bp_results$importarn_vips)

# models_stability_plot_bootstrap(bp_results)
```

`models_stability_plot_plsda` *Models stability plot*

Description

Plot stability among models of the external cross validation

Usage

```
models_stability_plot_plsda(model)
```

Arguments

`model`	A nmr_data_analysis_model

Value

A plot of models stability

Examples

```
# Data analysis for a table of integrated peaks

## Generate an artificial nmr_dataset_peak_table:
### Generate artificial metadata:
num_samples <- 32 # use an even number in this example
num_peaks <- 20
metadata <- data.frame(
    NMRExperiment = as.character(1:num_samples),
    Condition = rep(c("A", "B"), times = num_samples / 2)
)

### The matrix with peaks
peak_means <- runif(n = num_peaks, min = 300, max = 600)
peak_sd <- runif(n = num_peaks, min = 30, max = 60)
peak_matrix <- mapply(function(mu, sd) rnorm(num_samples, mu, sd),
    mu = peak_means, sd = peak_sd
)
colnames(peak_matrix) <- paste0("Peak", 1:num_peaks)

## Artificial differences depending on the condition:
peak_matrix[metadata$Condition == "A", "Peak2"] <-
    peak_matrix[metadata$Condition == "A", "Peak2"] + 70

peak_matrix[metadata$Condition == "A", "Peak6"] <-
    peak_matrix[metadata$Condition == "A", "Peak6"] - 60

### The nmr_dataset_peak_table
peak_table <- new_nmr_dataset_peak_table(
    peak_table = peak_matrix,
    metadata = list(external = metadata)
)

methodology <- plsda_auroc_vip_method(ncomp = 3)
model <- nmr_data_analysis(
    peak_table,
    y_column = "Condition",
    identity_column = NULL,
    external_val = list(iterations = 3, test_size = 0.25),
    internal_val = list(iterations = 3, test_size = 0.25),
    data_analysis_method = methodology
)

# models_stability_plot_plsda(model)
```

`new_nmr_dataset` *Create an nmr_dataset object*

Description

Create an nmr_dataset object

Usage

```
new_nmr_dataset(metadata, data_fields, axis)
```

Arguments

metadata	A named list of data frames
data_fields	A named list. Check the examples
axis	A list. Check the examples

Value

Create an nmr_dataset object

Create an nmr_dataset object

See Also

Other class helper functions: format.nmr_dataset(), format.nmr_dataset_1D(), format.nmr_dataset_peak_table(), is.nmr_dataset_1D(), is.nmr_dataset_peak_table(), new_nmr_dataset_1D(), new_nmr_dataset_peak_table(), print.nmr_dataset(), print.nmr_dataset_1D(), print.nmr_dataset_peak_table(), validate_nmr_dataset(), validate_nmr_dataset_family(), validate_nmr_dataset_peak_table()

Examples

```
#
metadata_1D <- list(external = data.frame(NMRExperiment = c("10", "20")))
# Sample 10 and Sample 20 can have different lengths (due to different setups)
data_fields_1D <- list(data_1r = list(runif(16), runif(32)))
# Each sample has its own axis list, with one element (because this example is 1D)
axis_1D <- list(list(1:16), list(1:32))
my_1D_data <- new_nmr_dataset(metadata_1D, data_fields_1D, axis_1D)

# Example for 2D samples
metadata_2D <- list(external = data.frame(NMRExperiment = c("11", "21")))
data_fields_2D <- list(data_2rr = list(matrix(runif(16 * 3), nrow = 16, ncol = 3),
    runif(32 * 3),
    nrow = 32, ncol = 3
))
# Each sample has its own axis list, with one element (because this example is 1D)
axis_2D <- list(list(1:16, 1:3), list(1:32, 1:3))
my_2D_data <- new_nmr_dataset(metadata_2D, data_fields_2D, axis_2D)
```

`new_nmr_dataset_1D` *Creates a new 1D nmr_dataset object from scratch*

Description

Creates a new 1D nmr_dataset object from scratch

Usage

```
new_nmr_dataset_1D(ppm_axis, data_1r, metadata)
```

Arguments

`ppm_axis`	A numeric vector with the ppm values for the columns of data_1r
`data_1r`	A numeric matrix with one NMR spectrum on each row
`metadata`	A list of data frames with at least the `NMRExperiment` column

Value

Creates a new 1D nmr_dataset object from scratch

See Also

Other class helper functions: `format.nmr_dataset()`, `format.nmr_dataset_1D()`, `format.nmr_dataset_peak_table()`, `is.nmr_dataset_1D()`, `is.nmr_dataset_peak_table()`, `new_nmr_dataset()`, `new_nmr_dataset_peak_table()`, `print.nmr_dataset()`, `print.nmr_dataset_1D()`, `print.nmr_dataset_peak_table()`, `validate_nmr_dataset()`, `validate_nmr_dataset_family()`, `validate_nmr_dataset_peak_table()`

Examples

```
# Create a random spectra matrix
nsamp <- 12
npoints <- 20
dummy_ppm_axis <- seq(from = 0.2, to = 10, length.out = npoints)
dummy_spectra_matrix <- matrix(runif(nsamp * npoints), nrow = nsamp, ncol = npoints)
metadata <- list(external = data.frame(
    NMRExperiment = paste0("Sample", 1:12),
    DummyClass = c("a", "b")
))
dummy_nmr_dataset_1D <- new_nmr_dataset_1D(
    ppm_axis = dummy_ppm_axis,
    data_1r = dummy_spectra_matrix,
    metadata = metadata
)
```

`new_nmr_dataset_peak_table` *Creates a new nmr_dataset_peak_table object from scratch*

Description

Creates a new nmr_dataset_peak_table object from scratch

Usage

```
new_nmr_dataset_peak_table(peak_table, metadata)
```

Arguments

`peak_table`	A numeric matrix with one NMR spectrum on each row
`metadata`	A list of data frames with at least the `NMRExperiment` column

Value

Creates a new nmr_dataset_peak_table object from scratch

See Also

Other class helper functions: `format.nmr_dataset()`, `format.nmr_dataset_1D()`, `format.nmr_dataset_peak_table()`, `is.nmr_dataset_1D()`, `is.nmr_dataset_peak_table()`, `new_nmr_dataset()`, `new_nmr_dataset_1D()`, `print.nmr_dataset()`, `print.nmr_dataset_1D()`, `print.nmr_dataset_peak_table()`, `validate_nmr_dataset()`, `validate_nmr_dataset_family()`, `validate_nmr_dataset_peak_table()`

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
meta <- file.path(dir_to_demo_dataset, "dummy_metadata.xlsx")
metadata <- readxl::read_excel(meta, sheet = 1)
dataset_1D <- nmr_meta_add(dataset_1D, metadata = metadata, by = "NMRExperiment")
metadata <- list(external = dataset_1D[["metadata"]][["external"]])
peak_table <- nmr_data(dataset_1D)
new <- new_nmr_dataset_peak_table(peak_table, metadata)
```

`nmr_align` *Align NMR spectra*

Description

This function is based on speaq::dohCluster.

Usage

```
nmr_align(
  nmr_dataset,
  peak_data,
  NMRExp_ref = NULL,
  maxShift_ppm = 0.0015,
  acceptLostPeak = FALSE
)
```

Arguments

`nmr_dataset`	An nmr_dataset_1D
`peak_data`	The detected peak data given by nmr_detect_peaks.
`NMRExp_ref`	NMRExperiment of the reference to use for alignment
`maxShift_ppm`	The maximum shift allowed, in ppm
`acceptLostPeak`	This is an option for users, TRUE is the default value. If the users believe that all the peaks in the peak list are true positive, change it to FALSE.

Value

An nmr_dataset_1D, with the spectra aligned

See Also

Other alignment functions: `Pipelines`, `nmr_align_find_ref()`

Other peak alignment functions: `nmr_align_find_ref()`

nmr_align_find_ref *Find alignment reference*

Description

Find alignment reference

Usage

```
nmr_align_find_ref(nmr_dataset, peak_data)
```

Arguments

nmr_dataset	An nmr_dataset_1D
peak_data	The detected peak data given by nmr_detect_peaks.

Value

The NMRExperiment of the reference sample

See Also

Other alignment functions: Pipelines, nmr_align()

Other peak alignment functions: nmr_align()

nmr_autophase *Rephase 1D NMR data*

Description

Use phasing algorithms to rephase data in the spectral domain.

This function may improve autophasing processing from instrument vendors. It wraps the NMRphasing::NMRphasing() function, to automatically rephase spectra, allowing you to choose from a number of algorithms of which NLS, MPC_DANM and SPC_DANM are the most recent.

Rephasing should happen before any spectra interpolation.

Please use the all_components = TRUE when calling nmr_read_samples() in order to load the complex spectra and fix NMR phasing correctly.

Usage

```
nmr_autophase(
  dataset,
 method = c("NLS", "MPC_DANM", "MPC_EMP", "SPC_DANM", "SPC_EMP", "SPC_AAM", "SPC_DSM"),
  withBC = FALSE,
  ...
)
```

Arguments

dataset	An nmr_dataset object
method	The autophasing method. See NMRphasing::NMRphasing() for details.
withBC	NMRphasing::NMRphasing may perform a baseline correction using modified polynomial fitting. By default AlpsNMR offers other baseline estimation methods and better visualization of its effect, so AlpsNMR by default disables the baseline correction offered by NMRphasing.
...	Other parameters passed on to NMRphasing::NMRphasing().

Value

A (hopefully better phased) nmr_dataset object, with updated real and imaginary parts.

Examples

```
if (requireNamespace("NMRphasing", quietly=TRUE)) {
  # Helpers to create a dataset:
  lorentzian <- function(x, x0, gamma, A) {
    A * (1 / (pi * gamma)) * ((gamma^2) / ((x - x0)^2 + gamma^2))
  }
  x <- seq(from=1, to=2, length.out = 300)
  y <- lorentzian(x, 1.3, 0.01, 1) + lorentzian(x, 1.6, 0.01, 1)
  dataset <- new_nmr_dataset(
    metadata = list(external = data.frame(NMRExperiment = "10")),
    data_fields = list(data_1r = list(y)),
    axis = list(list(x))
  )
  # Autophase, interpolate and plot:
  dataset <- nmr_autophase(dataset, method = "NLS")
  dataset <- nmr_interpolate_1D(dataset, axis = c(min = 1, max = 2, by = 0.01))
  plot(dataset)
}
```

---

nmr_baseline_estimation

Estimate the baseline on an nmr_dataset_1D object, using baseline::baseline.als.

Description

Estimate the baseline on an nmr_dataset_1D object, using baseline::baseline.als.

Usage

```
nmr_baseline_estimation(nmr_dataset, lambda = 9, p = 0.05, maxit = 20)
```

Arguments

nmr_dataset	An nmr_dataset_1D.
lambda	2nd derivative constraint
p	Weighting of positive residuals
maxit	Maximum number of iterations

Value

The same nmr_dataset_1D object with the data_1r_baseline element.

See Also

baseline::baseline.als

Other baseline removal functions: nmr_baseline_removal()

Examples

```
dataset_1D <- nmr_dataset_load(system.file("extdata", "nmr_dataset.rds", package = "AlpsNMR"))
dataset_1D <- nmr_baseline_estimation(dataset_1D, lambda = 9, p = 0.01)
```

---

nmr_baseline_removal *Baseline Removal NMR*

Description

Removes the baseline on an nmr_dataset_1D object, using baseline::baseline.als.

Usage

```
nmr_baseline_removal(nmr_dataset, lambda = 6, p = 0.05, maxit = 20)
```

Arguments

nmr_dataset	An nmr_dataset_1D.
lambda	2nd derivative constraint
p	Weighting of positive residuals
maxit	Maximum number of iterations

Value

The same nmr_dataset_1D object after baseline removal.

See Also

baseline::baseline.als

Other baseline removal functions: nmr_baseline_estimation()

Examples

```
dataset_1D <- nmr_dataset_load(system.file("extdata", "nmr_dataset.rds", package = "AlpsNMR"))
dataset_no_base_line <- nmr_baseline_removal(dataset_1D, lambda = 6, p = 0.01)
```

`nmr_baseline_threshold` *Threshold estimation for peak detection*

Description

Estimates the threshold value for peak detection on an nmr_dataset_1D object by examining a range without peaks, by default the 9.5 - 10 ppm range.

Usage

```
nmr_baseline_threshold(
  nmr_dataset,
  range_without_peaks = c(9.5, 10),
  method = c("mean3sd", "median3mad")
)
```

Arguments

`nmr_dataset`	An nmr_dataset_1D.
`range_without_peaks`	A vector with two doubles describing a range without peaks suitable for baseline detection
`method`	Either "mean3sd" or the more robust "median3mad". See the details.

Details

Two methods can be used:

- "mean3sd": The mean3sd method computes the mean and the standard deviation of each spectrum in the 9.5 - 10 ppm range. The mean spectrum and the mean standard deviation are both vectors of length equal to the number of points in the given range. The mean of the mean spectrum the noise. The threshold is defined as `center + 3 dispersion`, and it is one single threshold for the whole dataset. This is the default for backwards compatibility.
- "median3mad": First we take the data matrix. If we have estimated a baseline already, subtract it. In the defined region without peaks, estimate the median of each sample and its median absolute deviation. Return a vector of length equal to the number of samples with the `median+3mad` for each sample. This is a new more robust method.

Value

Numerical. A threshold value in intensity below that no peak is detected.

See Also

Other peak detection functions: Pipelines, nmr_detect_peaks(), nmr_detect_peaks_plot(), nmr_detect_peaks_plot_overview(), nmr_detect_peaks_tune_snr(), nmr_identify_regions_blood(), nmr_identify_regions_cell(), nmr_identify_regions_urine(), nmr_integrate_regions()

Examples

```
ppm_axis <- seq(from = 0, to = 10, length.out = 1000)
data_1r <- matrix(runif(1000, 0, 10), nrow = 1) + 100
dataset_1D <- new_nmr_dataset_1D(
    ppm_axis = ppm_axis,
    data_1r = data_1r,
    metadata = list(external=data.frame(NMRExperiment = "10"))
)
bl_threshold <- nmr_baseline_threshold(dataset_1D, range_without_peaks = c(9.5,10))
```

---

nmr_baseline_threshold_plot

Plot the baseline thresholds

Description

If you have a lot of samples you can make the plot window bigger (or use " ```{r fig.height=10, fig.width=10}" in notebooks), or choose some NMRExperiments.

Usage

```
nmr_baseline_threshold_plot(
  nmr_dataset,
  thresholds,
  NMRExperiment = "all",
  chemshift_range = c(9.5, 10),
  ...
)
```

Arguments

nmr_dataset	An nmr_dataset_1D object
thresholds	A named vector. The values are baseline thresholds. The names are NMRExperiments.
NMRExperiment	The NMRExperiments to plot (Use "all" to plot all of them)
chemshift_range	The range to plot, as a first check use the range_without_peaks from nmr_baseline_threshold
...	arguments passed to ggplot2::aes (or to ggplot2::aes_string, being deprecated).

Value

A plot.

Examples

```
ppm_axis <- seq(from = 0, to = 10, length.out = 1000)
data_1r <- matrix(runif(1000, 0, 10), nrow = 1) + 100
dataset_1D <- new_nmr_dataset_1D(
    ppm_axis = ppm_axis,
    data_1r = data_1r,
    metadata = list(external=data.frame(NMRExperiment = "10"))
)
bl_threshold <- nmr_baseline_threshold(dataset_1D, range_without_peaks = c(9.5,10))
baselineThresh <- nmr_baseline_threshold(dataset_1D)
nmr_baseline_threshold_plot(dataset_1D, bl_threshold)
```

---

nmr_batman *Batman helpers*

Description

Batman helpers

Usage

```
nmr_batman_write_options(
  bopts,
  batman_dir = "BatmanInput",
  filename = "batmanOptions.txt"
)

nmr_batman_export_dataset(
  nmr_dataset,
  batman_dir = "BatmanInput",
  filename = "NMRdata.txt"
)

nmr_batman_multi_data_user_hmdb(
  batman_dir = "BatmanInput",
  filename = "multi_data_user.csv"
)

nmr_batman_multi_data_user(
  multiplet_table,
  batman_dir = "BatmanInput",
  filename = "multi_data_user.csv"
)
```

```
nmr_batman_metabolites_list(
  metabolite_names,
  batman_dir = "BatmanInput",
  filename = "metabolitesList.csv"
)
```

Arguments

`bopts`	Batman options
`batman_dir`	Batman input directorye
`filename`	Filename to use, inside `batman_dir`
`nmr_dataset`	An nmr_dataset_1D object
`multiplet_table`	A data frame, like the hmdb dataset
`metabolite_names`	A character vector of the metabolite names to consider

Value

These are helper functions to make Batman tests easier

See Also

Other batman functions: `nmr_batman_options()`

Examples

```
bopts <- nmr_batman_options()
# nmr_batman_write_options(bopts)

dataset_1D <- nmr_dataset_load(system.file("extdata", "nmr_dataset.rds", package = "AlpsNMR"))
# nmr_batman_export_dataset(dataset_1D)

message("Use of multi_data_user_hmdb")
# multi_data_user_hmdb <- nmr_batman_multi_data_user_hmdb()
hmdb <- NULL
# utils::data("hmdb", package = "AlpsNMR", envir = environment())
# hmdb <- nmr_batman_multi_data_user(hmbd)

metabolite_names <- c("alanine", "glucose")
# metabolite_names <- nmr_batman_metabolites_list(metabolite_names)
```

nmr_batman_options *Batman Options helper*

Description

Batman Options helper

Usage

```
nmr_batman_options(
 ppmRange = matrix(c(3, 3.1, 3.6, 3.7, 3.9, 4, 4, 4.1, 6.95, 7.05, 7.6, 7.7, 7.8, 7.9),
    ncol = 2, byrow = TRUE),
 specNo = "1",
 paraProc = 4L,
 negThresh = -0.5,
 scaleFac = 1e+06,
 downSamp = 1,
 hiresFlag = 1,
 randSeed = 100025L,
 nItBurnin = 200L,
 nItPostBurnin = 5000L,
 multFile = 2L,
 thinning = 50L,
 cfeFlag = 0,
 nItRerun = 5000L,
 startTemp = 1000,
 specFreq = 600,
 a = 1e-05,
 b = 1e-09,
 muMean = 1.1,
 muVar = 0.2,
 muVar_prop = 0.002,
 nuMVar = 0.0025,
 nuMVarProp = 0.1,
 tauMean = -0.05,
 tauPrec = 2,
 rdelta = 0.02,
 csFlag = 0
)
```

Arguments

ppmRange	Range of ppm to process
specNo	Index of spectra to process
paraProc	Number of cores to use
negThresh	Truncation threshold for negative intensities

scaleFac	Divide each spectrum by this number
downSamp	Decimate each spectrum by this factor
hiresFlag	Keep High Resolution deconvolved spectra
randSeed	A random seed
nItBurnin	Number of burn-in iterations
nItPostBurnin	Number of iterations after burn-in
multFile	Multiplet file (integer)
thinning	Save MCMC state every thinning iterations
cfeFlag	Same concentration for all spectra (fixed effect)
nItRerun	Number of iterations for a batman rerun
startTemp	Start temperature
specFreq	NMR Spectrometer frequency
a	Shape parameter for the gamma distribution (used for lambda, the precision)
b	Rate distribution parameter for the gamma distribution (used for lambda, the precision)
muMean	Peak width mean in ln(Hz)
muVar	Peak width variance in ln(Hz)
muVar_prop	Peak width proposed variance in ln(Hz)
nuMVar	Peak width metabolite variance in ln(Hz)
nuMVarProp	Peak width metabolite proposed variance in ln(Hz)
tauMean	mean of the prior on tau
tauPrec	inverse of variance of prior on tau
rdelta	Truncation of the prior on peak shift (ppm)
csFlag	Specify chemical shift for each multiplet in each spectrum? (chemShiftperSpectra.csv file)

Value

A batman_options object with the Batman Options

See Also

Other batman functions: nmr_batman

Examples

```
bopts <- nmr_batman_options()
```

nmr_build_peak_table *Build a peak table from the clustered peak list*

Description

Build a peak table from the clustered peak list

Usage

```
nmr_build_peak_table(peak_data, dataset = NULL)
```

Arguments

peak_data	A peak list, with the cluster column
dataset	A nmr_dataset_1D object, to get the metadata

Value

An nmr_dataset_peak_table, containing the peak table and the annotations

Examples

```
peak_data <- data.frame(
    NMRExperiment = c("10", "10", "20", "20"),
    peak_id = paste0("Peak", 1:4),
    ppm = c(1, 2, 1.1, 2.1),
    area = c(10, 20, 12, 22)
)
clustering_result <- nmr_peak_clustering(peak_data, num_clusters = 2)
peak_data <- clustering_result$peak_data
peak_table <- nmr_build_peak_table(peak_data)
stopifnot(ncol(peak_table) == 2)
```

nmr_data *Set/Return the full spectra matrix*

Description

Set/Return the full spectra matrix

Usage

```
nmr_data(nmr_dataset, ...)

## S3 method for class 'nmr_dataset_1D'
nmr_data(nmr_dataset, what = "data_1r", ...)

nmr_data(nmr_dataset, ...) <- value

## S3 replacement method for class 'nmr_dataset_1D'
nmr_data(nmr_dataset, what = "data_1r", ...) <- value
```

Arguments

nmr_dataset	An object from the nmr_dataset_family to get the raw data from
...	Passed on to methods for compatibility
what	What data do we want to get (default: data_1r)
value	A matrix

Value

a matrix

The given nmr_dataset

See Also

Other import/export functions: Pipelines, files_to_rDolphin(), load_and_save_functions, nmr_meta_export(), nmr_read_bruker_fid(), nmr_read_samples(), nmr_zip_bruker_samples(), save_files_to_rDolphin(), save_profiling_output(), to_ChemoSpec()

Examples

```
dataset_rds <- system.file("extdata", "nmr_dataset.rds", package = "AlpsNMR")
dataset_1D <- nmr_dataset_load(dataset_rds)
dataset_data <- nmr_data(dataset_1D)
dataset_rds <- system.file("extdata", "nmr_dataset.rds", package = "AlpsNMR")
dataset_1D <- nmr_dataset_load(dataset_rds)
dataset_1D_data <- nmr_data(dataset_1D)
```

---

nmr_dataset *nmr_dataset (S3 class)*

Description

An nmr_dataset represents a set of NMR samples. It is defined as an S3 class, and it can be treated as a regular list.

Details

It currently has the following elements:

- `metadata`: A list of data frames. Each data frame contains metadata of a given area (acquisition parameters, preprocessing parameters, general sample information...)
- `axis`: A list with length equal to the dimensionality of the data. For 1D spectra it is a list with a numeric vector
- `data_*`: Data arrays with the actual spectra. The first index represents the sample, the rest of the indices match the length of each `axis`. Typically `data_1r` is a matrix with one sample on each row and the chemical shifts in the columns.
- `num_samples`: The number of samples in the dataset

See Also

Functions to save and load these objects

Other AlpsNMR dataset objects: `nmr_dataset_family`

Examples

```
metadata_1D <- list(external = data.frame(NMRExperiment = c("10", "20")))
# Sample 10 and Sample 20 can have different lengths (due to different setups)
data_fields_1D <- list(data_1r = list(runif(16), runif(32)))
# Each sample has its own axis list, with one element (because this example is 1D)
axis_1D <- list(list(1:16), list(1:32))
my_1D_data <- new_nmr_dataset(metadata_1D, data_fields_1D, axis_1D)
```

`nmr_dataset_1D` *nmr_dataset_1D (S3 class)*

Description

An `nmr_dataset_1D` represents a set of 1D interpolated NMR samples. It is defined as an S3 class, and it can be treated as a regular list.

Details

It currently has the following elements:

- `metadata`: A list of data frames. Each data frame contains metadata of a given area (acquisition parameters, preprocessing parameters, general sample information...)
- `axis`: A numeric vector with the chemical shift axis in ppm.
- `data_1r`: A matrix with one sample on each row and the chemical shifts in the columns.

Examples

```
# Create a random spectra matrix
nsamp <- 12
npoints <- 20
dummy_ppm_axis <- seq(from = 0.2, to = 10, length.out = npoints)
dummy_spectra_matrix <- matrix(runif(nsamp * npoints), nrow = nsamp, ncol = npoints)
metadata <- list(external = data.frame(
    NMRExperiment = paste0("Sample", 1:12),
    DummyClass = c("a", "b")
))
dummy_nmr_dataset_1D <- new_nmr_dataset_1D(
    ppm_axis = dummy_ppm_axis,
    data_1r = dummy_spectra_matrix,
    metadata = metadata
)
```

---

nmr_dataset_family *nmr_dataset like objects (S3 classes)*

Description

The AlpsNMR package defines and uses several objects to manage NMR Data.

Details

These objects share some structure and functions, so it makes sense to have an abstract class to ensure that the shared structures are compatible

See Also

Functions to save and load these objects

Other AlpsNMR dataset objects: `nmr_dataset`

---

nmr_dataset_peak_table *nmr_dataset_peak_table (S3 class)*

Description

An `nmr_dataset_peak_table` represents a peak table with metadata. It is defined as an S3 class, and it can be treated as a regular list.

Usage

```
## S3 method for class 'nmr_dataset_peak_table'
as.data.frame(x, ...)
```

Arguments

x	An nmr_dataset_peak_table object,
...	ignored

Details

- `metadata`: A list of data frames. Each data frame contains metadata. Usually the list only has one data frame named "external".
- `peak_table`: A matrix with one sample on each row and the peaks in the columns

Value

A data frame with the sample metadata and the peak table

Methods (by generic)

- `as.data.frame(nmr_dataset_peak_table)`: Convert to a data frame

`nmr_dataset_peak_table_to_SummarizedExperiment` *Export nmr_dataset_peak_table to SummarizedExperiment*

Description

Export nmr_dataset_peak_table to SummarizedExperiment

Usage

```
nmr_dataset_peak_table_to_SummarizedExperiment(nmr_peak_table)
```

Arguments

`nmr_peak_table`	An nmr_dataset_peak_table object

Value

SummarizedExperiment object (unmodified)

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
meta <- file.path(dir_to_demo_dataset, "dummy_metadata.xlsx")
metadata <- readxl::read_excel(meta, sheet = 1)
dataset_1D <- nmr_meta_add(dataset_1D, metadata = metadata, by = "NMRExperiment")
metadata <- list(external = dataset_1D[["metadata"]][["external"]])
peak_table <- nmr_data(dataset_1D)
```

```
nmr_peak_table <- new_nmr_dataset_peak_table(peak_table, metadata)
se <- nmr_dataset_peak_table_to_SummarizedExperiment(nmr_peak_table)
```

`nmr_data_1r_to_SummarizedExperiment` *Export 1D NMR data to SummarizedExperiment*

Description

Export 1D NMR data to SummarizedExperiment

Usage

```
nmr_data_1r_to_SummarizedExperiment(nmr_dataset)
```

Arguments

`nmr_dataset`	An nmr_dataset_1D object

Value

SummarizedExperiment An SummarizedExperiment object (unmodified)

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
se <- nmr_data_1r_to_SummarizedExperiment(dataset_1D)
```

`nmr_data_analysis` *Data analysis*

Description

Data analysis on AlpsNMR can be performed on both nmr_dataset_1D full spectra as well as nmr_dataset_peak_table peak tables.

Usage

```
nmr_data_analysis(
  dataset,
  y_column,
  identity_column,
  external_val,
  internal_val,
  data_analysis_method,
  .enable_parallel = TRUE
)
```

Arguments

`dataset`	An nmr_dataset_family object
`y_column`	A string with the name of the y column (present in the metadata of the dataset)
`identity_column`	`NULL` or a string with the name of the identity column (present in the metadata of the dataset).
`external_val, internal_val`	A list with two elements: `iterations` and `test_size`. See random_subsampling for further details
`data_analysis_method`	An nmr_data_analysis_method object
`.enable_parallel`	Set to `FALSE` to disable parallellization.

Details

The workflow consists of a double cross validation strategy using random subsampling for splitting into train and test sets. The classification model and the metric to choose the best model can be customized (see `new_nmr_data_analysis_method()`), but for now only a PLSDA classification model with a best area under ROC curve metric is implemented (see the examples here and plsda_auroc_vip_method)

Value

A list with the following elements:

- `train_test_partitions`: A list with the indices used in train and test on each of the cross-validation iterations
- `inner_cv_results`: The output returned by `train_evaluate_model` on each inner cross-validation
- `inner_cv_results_digested`: The output returned by `choose_best_inner`.
- `outer_cv_results`: The output returned by `train_evaluate_model` on each outer cross-validation
- `outer_cv_results_digested`: The output returned by `train_evaluate_model_digest_outer`.

Examples

```
# Data analysis for a table of integrated peaks

## Generate an artificial nmr_dataset_peak_table:
### Generate artificial metadata:
num_samples <- 32 # use an even number in this example
num_peaks <- 20
metadata <- data.frame(
    NMRExperiment = as.character(1:num_samples),
    Condition = rep(c("A", "B"), times = num_samples / 2)
)
```

```
### The matrix with peaks
peak_means <- runif(n = num_peaks, min = 300, max = 600)
peak_sd <- runif(n = num_peaks, min = 30, max = 60)
peak_matrix <- mapply(function(mu, sd) rnorm(num_samples, mu, sd),
    mu = peak_means, sd = peak_sd
)
colnames(peak_matrix) <- paste0("Peak", 1:num_peaks)

## Artificial differences depending on the condition:
peak_matrix[metadata$Condition == "A", "Peak2"] <-
    peak_matrix[metadata$Condition == "A", "Peak2"] + 70

peak_matrix[metadata$Condition == "A", "Peak6"] <-
    peak_matrix[metadata$Condition == "A", "Peak6"] - 60

### The nmr_dataset_peak_table
peak_table <- new_nmr_dataset_peak_table(
    peak_table = peak_matrix,
    metadata = list(external = metadata)
)

## We will use a double cross validation, splitting the samples with random
## subsampling both in the external and internal validation.
## The classification model will be a PLSDA, exploring at maximum 3 latent
## variables.
## The best model will be selected based on the area under the ROC curve
methodology <- plsda_auroc_vip_method(ncomp = 3)
model <- nmr_data_analysis(
    peak_table,
    y_column = "Condition",
    identity_column = NULL,
    external_val = list(iterations = 3, test_size = 0.25),
    internal_val = list(iterations = 3, test_size = 0.25),
    data_analysis_method = methodology
)
## Area under ROC for each outer cross-validation iteration:
model$outer_cv_results_digested$auroc
## Rank Product of the Variable Importance in the Projection
## (Lower means more important)
sort(model$outer_cv_results_digested$vip_rankproducts)
```

`nmr_data_analysis_method`

Create method for NMR data analysis

Description

Create method for NMR data analysis

Usage

```
new_nmr_data_analysis_method(
  train_evaluate_model,
  train_evaluate_model_params_inner,
  choose_best_inner,
  train_evaluate_model_params_outer,
  train_evaluate_model_digest_outer
)
```

Arguments

`train_evaluate_model`

A function. The `train_evaluate_model` must have the following signature:

```
function(x_train, y_train, identity_train, x_test, y_test, identity_test, ...)
```

The `x_train` and `y_train` (and their test counterparts) are self-explanatory.

The `identity_` arguments are expected to be factors. They can be used for instance with a callback that uses mixOmics::plsda in a `multilevel` approach for longitudinal studies. In those studies the `identity` would be an identifier of the subject.

The `...` arguments are free to be defined for each `train_evaluate_model`.

`train_evaluate_model_params_inner`, `train_evaluate_model_params_outer`

A list with additional arguments to pass to `train_evaluate_model` either in the inner cv loop or in the outer cv loop.

`choose_best_inner`

A function with a single argument:

```
function(inner_cv_results)
```

The argument is a list of `train_evaluate_model` outputs. The return value of must be a list with at least an element named `train_evaluate_model_args`. `train_evaluate_model_args` must be a named list.

- Each element must be named as one of the `train_evaluate_model` arguments.
- Each element must be a vector as long as the number of outer cross-validations.
- The values of each vector must be the values that the `train_evaluate_model` argument must take on each outer cross-validation iteration Additional list elements can be returned and will be given back to the user

`train_evaluate_model_digest_outer`

A function with a single argument:

function(outer_cv_results)

The argument is a list of `train_evaluate_model` outputs in outer cross-validation. The return value is returned by `nmr_data_analysis`

Value

An object encapsulating the method dependent functions that can be used with nmr_data_analysis

`nmr_detect_peaks` *Peak detection for NMR*

Description

The function detects peaks on an nmr_dataset_1D object, using speaq::detectSpecPeaks. `detectSpecPeaks` divides the whole spectra into smaller segments and uses MassSpecWavelet::peakDetectionCWT for peak detection.

Usage

```
nmr_detect_peaks(
  nmr_dataset,
  nDivRange_ppm = 0.1,
  scales = seq(1, 16, 2),
  baselineThresh = NULL,
  SNR.Th = 3,
  range_without_peaks = c(9.5, 10),
  fit_lorentzians = FALSE,
  verbose = FALSE
)
```

Arguments

`nmr_dataset`	An nmr_dataset_1D.
`nDivRange_ppm`	Segment size, in ppms, to divide the spectra and search for peaks.
`scales`	The parameter of peakDetectionCWT function of MassSpecWavelet package, look it up in the original function.
`baselineThresh`	All peaks with intensities below the thresholds are excluded. Either: • A numeric vector of length the number of samples. Each number is a threshold for that sample • A single number. All samples use this number as baseline threshold. • `NULL`. If that's the case, a default function is used (`nmr_baseline_threshold()`), which assumes that there is no signal in the region 9.5-10 ppm.
`SNR.Th`	The parameter of peakDetectionCWT function of MassSpecWavelet package, look it up in the original function. If you set -1, the function will itself recompute this value.
`range_without_peaks`	A numeric vector of length two with a region without peaks, only used when `baselineThresh = NULL`
`fit_lorentzians`	If `TRUE`, fit a lorentzian to each detected peak, to infer its inflection points. For now disabled for backwards compatibility.
`verbose`	Logical (`TRUE` or `FALSE`). Show informational messages, such as the estimated baseline

Details

Optionally afterwards, the peak apex and the peak inflection points are used to efficiently adjust a lorentzian to each peak, and compute the peak area and width, as well as the error of the fit. These peak features can be used afterwards to reject false detections.

Value

A data frame with the NMRExperiment, the sample index, the position in ppm and index and the peak intensity

See Also

nmr_align for peak alignment with the detected peak table

Peak_detection

Other peak detection functions: Pipelines, nmr_baseline_threshold(), nmr_detect_peaks_plot(), nmr_detect_peaks_plot_overview(), nmr_detect_peaks_tune_snr(), nmr_identify_regions_blood(), nmr_identify_regions_cell(), nmr_identify_regions_urine(), nmr_integrate_regions()

---

nmr_detect_peaks_plot *Plot peak detection results*

Description

Plot peak detection results

Usage

```
nmr_detect_peaks_plot(
  nmr_dataset,
  peak_data,
  NMRExperiment = NULL,
  peak_id = NULL,
  accepted_only = NULL,
  ...
)
```

Arguments

nmr_dataset	An nmr_dataset_1D.
peak_data	The peak table returned by nmr_detect_peaks
NMRExperiment	a single NMR experiment to plot
peak_id	A character vector. If given, plot only that peak id.
accepted_only	If peak_data contains a logical column named accepted, only those with accepted=TRUE will be counted. By default, accepted_only = TRUE, unless a peak_id is given
...	Arguments passed to plot.nmr_dataset_1D (chemshift_range, ...)

Value

Plot peak detection results

See Also

Peak_detection nmr_detect_peaks

Other peak detection functions: Pipelines, nmr_baseline_threshold(), nmr_detect_peaks(), nmr_detect_peaks_plot_overview(), nmr_detect_peaks_tune_snr(), nmr_identify_regions_blood(), nmr_identify_regions_cell(), nmr_identify_regions_urine(), nmr_integrate_regions()

Other peak detection functions: Pipelines, nmr_baseline_threshold(), nmr_detect_peaks(), nmr_detect_peaks_plot_overview(), nmr_detect_peaks_tune_snr(), nmr_identify_regions_blood(), nmr_identify_regions_cell(), nmr_identify_regions_urine(), nmr_integrate_regions()

---

nmr_detect_peaks_plot_overview

Overview of the peak detection results

Description

This plot allows to explore the performance of the peak detection across all the samples, by summarizing how many peaks are detected on each sample at each chemical shift range.

Usage

```
nmr_detect_peaks_plot_overview(
  peak_data,
  ppm_breaks = pretty(range(peak_data$ppm), n = 20),
  accepted_only = TRUE
)
```

Arguments

peak_data	The output of nmr_detect_peaks()
ppm_breaks	A numeric vector with the breaks that will be used to count the number of the detected peaks.
accepted_only	If peak_data contains a logical column named accepted, only those with accepted=TRUE will be counted.

Details

You can use this plot to find differences in the number of detected peaks across your dataset, and then use nmr_detect_peaks_plot() to have a finer look at specific samples and chemical shifts, and assess graphically that the peak detection results that you have are correct.

Value

A scatter plot, with samples on one axis and chemical shift bins in the other axis. The size of each dot represents the number of peaks found on a sample within a chemical shift range.

See Also

Peak_detection

Other peak detection functions: Pipelines, nmr_baseline_threshold(), nmr_detect_peaks(), nmr_detect_peaks_plot(), nmr_detect_peaks_tune_snr(), nmr_identify_regions_blood(), nmr_identify_regions_cell(), nmr_identify_regions_urine(), nmr_integrate_regions()

---

nmr_detect_peaks_plot_peaks

Plot multiple peaks from a peak list

Description

Plot multiple peaks from a peak list

Usage

```
nmr_detect_peaks_plot_peaks(
  nmr_dataset,
  peak_data,
  peak_ids,
  caption = paste("{peak_id}", "(NMRExp.\u00A0{NMRExperiment},",
    "\u03B3(ppb)\u00a0=\u00a0{gamma_ppb},", "\narea\u00a0=\u00a0{area},",
    "nrmse\u00a0=\u00a0{norm_rmse})")
)
```

Arguments

nmr_dataset	The nmr_dataset_1D object with the spectra
peak_data	A data frame, the peak list
peak_ids	The peak ids to plot
caption	The caption for each subplot

Value

A plot object

nmr_detect_peaks_tune_snr *Diagnose SNR threshold in peak detection*

Description

Diagnose SNR threshold in peak detection

Usage

```
nmr_detect_peaks_tune_snr(
  ds,
  NMRExperiment = NULL,
  SNR_thresholds = seq(from = 2, to = 6, by = 0.1),
  ...
)
```

Arguments

ds	An nmr_dataset_1D dataset
NMRExperiment	A string with the single NMRExperiment used explore the SNR thresholds. If not given, use the first one.
SNR_thresholds	A numeric vector with the SNR thresholds to explore
...	Arguments passed on to nmr_detect_peaks nmr_dataset An nmr_dataset_1D. nDivRange_ppm Segment size, in ppms, to divide the spectra and search for peaks. baselineThresh All peaks with intensities below the thresholds are excluded. Either: • A numeric vector of length the number of samples. Each number is a threshold for that sample • A single number. All samples use this number as baseline threshold. • NULL. If that's the case, a default function is used (nmr_baseline_threshold()), which assumes that there is no signal in the region 9.5-10 ppm. range_without_peaks A numeric vector of length two with a region without peaks, only used when baselineThresh = NULL fit_lorentzians If TRUE, fit a lorentzian to each detected peak, to infer its inflection points. For now disabled for backwards compatibility. verbose Logical (TRUE or FALSE). Show informational messages, such as the estimated baseline scales The parameter of peakDetectionCWT function of MassSpecWavelet package, look it up in the original function. SNR.Th The parameter of peakDetectionCWT function of MassSpecWavelet package, look it up in the original function. If you set -1, the function will itself re-compute this value.

Value

A list with the following elements:

- peaks_detected: A data frame with the columns from the nmr_detect_peaks output and an additional column SNR_threshold with the threshold used on each row.
- num_peaks_per_region: A summary of the peaks_detected table, with the number of peaks detected on each chemical shift region
- plot_num_peaks_per_region: A visual representation of num_peaks_per_region
- plot_spectrum_and_detections: A visual representation of the spectrum and the peaks detected with each SNR threshold. Use plotly::ggplotly or plot_interactive on this to zoom and explore the results.

See Also

nmr_detect_peaks

Other peak detection functions: Pipelines, nmr_baseline_threshold(), nmr_detect_peaks(), nmr_detect_peaks_plot(), nmr_detect_peaks_plot_overview(), nmr_identify_regions_blood(), nmr_identify_regions_cell(), nmr_identify_regions_urine(), nmr_integrate_regions()

---

nmr_exclude_region *Exclude region from samples*

Description

Excludes a given region (for instance to remove the water peak)

Usage

```
nmr_exclude_region(samples, exclude = list(water = c(4.7, 5)))

## S3 method for class 'nmr_dataset_1D'
nmr_exclude_region(samples, exclude = list(water = c(4.7, 5)))
```

Arguments

samples	An object
exclude	A list with regions to be removed Typically: exclude = list(water = c(4.7, 5.0))

Value

The same object, with the regions excluded

See Also

Other basic functions: nmr_normalize()

Examples

```
nmr_dataset <- nmr_dataset_load(system.file("extdata", "nmr_dataset.rds", package = "AlpsNMR"))
exclude_regions <- list(water = c(5.1, 4.5))
nmr_dataset <- nmr_exclude_region(nmr_dataset, exclude = exclude_regions)

nmr_dataset <- nmr_dataset_load(system.file("extdata", "nmr_dataset.rds", package = "AlpsNMR"))
exclude_regions <- list(water = c(5.1, 4.5))
nmr_dataset <- nmr_exclude_region(nmr_dataset, exclude = exclude_regions)
```

---

nmr_export_data_1r *Export 1D NMR data to a CSV file*

Description

Export 1D NMR data to a CSV file

Usage

```
nmr_export_data_1r(nmr_dataset, filename)
```

Arguments

nmr_dataset	An nmr_dataset_1D object
filename	The csv filename

Value

The nmr_dataset object (unmodified)

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
# nmr_export_data_1r(dataset_1D, "exported_nmr_dataset")
```

nmr_get_peak_distances *Compute peak to peak distances*

Description

Compute peak to peak distances

Usage

```
nmr_get_peak_distances(peak_data, same_sample_dist_factor = 3)
```

Arguments

peak_data	A peak list
same_sample_dist_factor	The distance between two peaks from the same sample are set to this factor multiplied by the maximum of all the peak distances

Value

A dist object with the peak2peak distances

Examples

```
peak_data <- data.frame(
    NMRExperiment = c("10", "10", "20", "20"),
    peak_id = paste0("Peak", 1:4),
    ppm = c(1, 2, 1.1, 3)
)
peak2peak_dist <- nmr_get_peak_distances(peak_data)
stopifnot(abs(as.numeric(peak2peak_dist) - c(6, 0.1, 2, 0.9, 1, 6)) < 1E-8)
```

nmr_identify_regions_blood *NMR peak identification (plasma/serum samples)*

Description

Identify given regions and return a data frame with plausible assignations in human plasma/serum samples.

Usage

```
nmr_identify_regions_blood(
  ppm_to_assign,
  num_proposed_compounds = 3,
  verbose = FALSE
)
```

Arguments

ppm_to_assign	A vector with the ppm regions to assign
num_proposed_compounds	set the number of proposed metabolites sorted by the number times reported in the HMDB: HMDB_blood.
verbose	Logical value. Set it to TRUE to print additional information

Value

a data frame with plausible assignations.

See Also

Other peak detection functions: Pipelines, nmr_baseline_threshold(), nmr_detect_peaks(), nmr_detect_peaks_plot(), nmr_detect_peaks_plot_overview(), nmr_detect_peaks_tune_snr(), nmr_identify_regions_cell(), nmr_identify_regions_urine(), nmr_integrate_regions()

Other peak integration functions: Pipelines, get_integration_with_metadata(), nmr_identify_regions_cell(), nmr_identify_regions_urine(), nmr_integrate_peak_positions(), nmr_integrate_regions()

Examples

```
# We identify regions from from the corresponding ppm storaged in a vector.
ppm_to_assign <- c(
    4.060960203, 3.048970634, 2.405935596,
    3.24146865, 0.990616851, 1.002075066, 0.955325548
)
identification <- nmr_identify_regions_blood(ppm_to_assign)
```

---

nmr_identify_regions_cell

NMR peak identification (cell samples)

Description

Identify given regions and return a data frame with plausible assignations in cell samples.

Usage

```
nmr_identify_regions_cell(
  ppm_to_assign,
  num_proposed_compounds = 3,
  verbose = FALSE
)
```

Arguments

ppm_to_assign	A vector with the ppm regions to assign
num_proposed_compounds	set the number of proposed metabolites in HMDB_cell.
verbose	Logical value. Set it to TRUE to print additional information

Value

a data frame with plausible assignations.

See Also

Other peak detection functions: Pipelines, nmr_baseline_threshold(), nmr_detect_peaks(), nmr_detect_peaks_plot(), nmr_detect_peaks_plot_overview(), nmr_detect_peaks_tune_snr(), nmr_identify_regions_blood(), nmr_identify_regions_urine(), nmr_integrate_regions()

Other peak integration functions: Pipelines, get_integration_with_metadata(), nmr_identify_regions_blood(), nmr_identify_regions_urine(), nmr_integrate_peak_positions(), nmr_integrate_regions()

Examples

```
# We identify regions from from the corresponding ppm storaged in a vector.
ppm_to_assign <- c(
    4.060960203, 3.048970634, 2.405935596,
    3.24146865, 0.990616851, 1.002075066, 0.955325548
)
identification <- nmr_identify_regions_cell(ppm_to_assign, num_proposed_compounds = 3)
```

---

nmr_identify_regions_urine

NMR peak identification (urine samples)

Description

Identify given regions and return a data frame with plausible assignations in human urine samples. The data frame contains the column "Bouatra_2013" showing if the proposed metabolite was reported in this publication as regular urinary metabolite.

Usage

```
nmr_identify_regions_urine(
  ppm_to_assign,
  num_proposed_compounds = 5,
  verbose = FALSE
)
```

Arguments

ppm_to_assign	A vector with the ppm regions to assign
num_proposed_compounds	set the number of proposed metabolites sorted by the number times reported in the HMDB: HMDB_urine.
verbose	Logical value. Set it to TRUE to print additional information

Value

a data frame with plausible assignations.

See Also

Other peak detection functions: Pipelines, nmr_baseline_threshold(), nmr_detect_peaks(), nmr_detect_peaks_plot(), nmr_detect_peaks_plot_overview(), nmr_detect_peaks_tune_snr(), nmr_identify_regions_blood(), nmr_identify_regions_cell(), nmr_integrate_regions()

Other peak integration functions: Pipelines, get_integration_with_metadata(), nmr_identify_regions_blood(), nmr_identify_regions_cell(), nmr_integrate_peak_positions(), nmr_integrate_regions()

Examples

```
# We identify regions from from the corresponding ppm storaged in a vector.
ppm_to_assign <- c(
    4.060960203, 3.048970634, 2.405935596,
    3.24146865, 0.990616851, 1.002075066, 0.955325548
)
identification <- nmr_identify_regions_urine(ppm_to_assign, num_proposed_compounds = 5)
```

---

nmr_integrate_peak_positions

Integrate peak positions

Description

The function allows the integration of a given ppm vector with a specific width.

Usage

```
nmr_integrate_peak_positions(
  samples,
  peak_pos_ppm,
  peak_width_ppm = 0.006,
  ...
)
```

Arguments

samples	A nmr_dataset object
peak_pos_ppm	The peak positions, in ppm
peak_width_ppm	The peak widths (or a single peak width for all peaks)
...	Arguments passed on to nmr_integrate_regions
	regions A named list. Each element of the list is a region, given as a named numeric vector of length two with the range to integrate. The name of the region will be the name of the column

Value

Integrate peak positions

See Also

Other peak integration functions: Pipelines, get_integration_with_metadata(), nmr_identify_regions_blood(), nmr_identify_regions_cell(), nmr_identify_regions_urine(), nmr_integrate_regions()

Other nmr_dataset_1D functions: [.nmr_dataset_1D(), format.nmr_dataset_1D(), get_integration_with_metadata is.nmr_dataset_1D(), nmr_integrate_regions(), nmr_meta_add(), nmr_meta_export(), nmr_meta_get(), nmr_meta_get_column(), nmr_ppm_resolution(), print.nmr_dataset_1D()

---

nmr_integrate_regions *Integrate regions*

Description

Integrate given regions and return a data frame with them

Usage

```
nmr_integrate_regions(samples, regions, ...)

## S3 method for class 'nmr_dataset_1D'
nmr_integrate_regions(
  samples,
  regions,
```

```
  fix_baseline = FALSE,
  excluded_regions_as_zero = FALSE,
  set_negative_areas_to_zero = FALSE,
  ...
)
```

Arguments

samples	A nmr_dataset object
regions	A named list. Each element of the list is a region, given as a named numeric vector of length two with the range to integrate. The name of the region will be the name of the column
...	Keep for compatibility
fix_baseline	A logical. If TRUE it removes the baseline. See details below
excluded_regions_as_zero	A logical. It determines the behaviour of the integration when integrating regions that have been excluded. If TRUE, it will treat those regions as zero. If FALSE (the default) it will return NA values. If fix_baseline is TRUE, then the region boundaries are used to estimate a baseline. The baseline is estimated "connecting the boundaries with a straight line". Only when the spectrum is above the baseline the area is integrated (negative contributions due to the baseline estimation are ignored).
set_negative_areas_to_zero	A logical. Ignored if fix_baseline is FALSE. When set to TRUE negative areas are set to zero.

Value

An nmr_dataset_peak_table object

See Also

Other peak detection functions: Pipelines, nmr_baseline_threshold(), nmr_detect_peaks(), nmr_detect_peaks_plot(), nmr_detect_peaks_plot_overview(), nmr_detect_peaks_tune_snr(), nmr_identify_regions_blood(), nmr_identify_regions_cell(), nmr_identify_regions_urine()

Other peak integration functions: Pipelines, get_integration_with_metadata(), nmr_identify_regions_blood(), nmr_identify_regions_cell(), nmr_identify_regions_urine(), nmr_integrate_peak_positions()

Other nmr_dataset_1D functions: [.nmr_dataset_1D(), format.nmr_dataset_1D(), get_integration_with_metadata is.nmr_dataset_1D(), nmr_integrate_peak_positions(), nmr_meta_add(), nmr_meta_export(), nmr_meta_get(), nmr_meta_get_column(), nmr_ppm_resolution(), print.nmr_dataset_1D()

Examples

```
# Creating a dataset
dataset <- new_nmr_dataset_1D(
    ppm_axis = 1:10,
    data_1r = matrix(sample(0:99, replace = TRUE), nrow = 10),
    metadata = list(external = data.frame(NMRExperiment = c(
```

```
        "10",
        "20", "30", "40", "50", "60", "70", "80", "90", "100"
    )))
)

# Integrating selected regions
peak_table_integration <- nmr_integrate_regions(
    samples = dataset,
    regions = list(ppm = c(2, 5))
)

# Creating a dataset
dataset <- new_nmr_dataset_1D(
    ppm_axis = 1:10,
    data_1r = matrix(sample(0:99, replace = TRUE), nrow = 10),
    metadata = list(external = data.frame(NMRExperiment = c(
        "10",
        "20", "30", "40", "50", "60", "70", "80", "90", "100"
    )))
)

# Integrating selected regions
peak_table_integration <- nmr_integrate_regions(
    samples = dataset,
    regions = list(ppm = c(2, 5)),
    fix_baseline = FALSE
)
```

`nmr_interpolate_1D` *Interpolate a set of 1D NMR Spectra*

Description

Interpolate a set of 1D NMR Spectra

Usage

```
nmr_interpolate_1D(samples, axis = c(min = 0.2, max = 10, by = 8e-04))

## S3 method for class 'nmr_dataset'
nmr_interpolate_1D(samples, axis = c(min = 0.2, max = 10, by = 8e-04))
```

Arguments

`samples`	An NMR dataset
`axis`	The ppm axis range and optionally the ppm step. Set it to `NULL` for autodetection

Value

Interpolate a set of 1D NMR Spectra

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))

dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
```

---

nmr_meta_add *Add metadata to an nmr_dataset object*

Description

This is useful to add metadata to datasets that can be later used for plotting spectra or further analysis (PCA...).

Usage

```
nmr_meta_add(nmr_data, metadata, by = "NMRExperiment")

nmr_meta_add_tidy_excel(nmr_data, excel_file)
```

Arguments

nmr_data	an nmr_dataset_family object
metadata	A data frame with metadata to add
by	A column name of both the nmr_data$metadata$external and the metadata data.frame. If you want to merge two columns with different headers you can use a named character vector c("NMRExperiment" = "ExperimentNMR") where the left side is the column name of the nmr_data$metadata$external and the right side is the column name of the metadata data frame.
excel_file	Path to a tidy Excel file name. The Excel can consist of multiple sheets, that are added sequentially. The first column of the first sheet MUST be named as one of the metadata already present in the dataset, typically will be "NMRExperiment". The rest of the columns of the first sheet can be named at will. Similary, the first column of the second sheet must be named as one of the metadata already present in the dataset, typically "NMRExperiment" or any of the columns of the first sheet. The rest of the columns of the second sheet can be named at will. See the package vignette for an example.

Value

The nmr_dataset_family object with the added metadata

See Also

Other metadata functions: Pipelines, nmr_meta_export(), nmr_meta_get(), nmr_meta_get_column(), nmr_meta_groups()

Other nmr_dataset functions: nmr_meta_export(), nmr_meta_get(), nmr_meta_get_column()

Other nmr_dataset_1D functions: [.nmr_dataset_1D(), format.nmr_dataset_1D(), get_integration_with_metadata is.nmr_dataset_1D(), nmr_integrate_peak_positions(), nmr_integrate_regions(), nmr_meta_export(), nmr_meta_get(), nmr_meta_get_column(), nmr_ppm_resolution(), print.nmr_dataset_1D()

Other nmr_dataset_peak_table functions: nmr_meta_export(), nmr_meta_get(), nmr_meta_get_column()

Examples

```
# Load a demo dataset with four samples:
dataset <- system.file("dataset-demo", package = "AlpsNMR")
nmr_dataset <- nmr_read_samples_dir(dataset)

# At first we just have the NMRExperiment column
nmr_meta_get(nmr_dataset, groups = "external")
# Get a table with NMRExperiment -> SubjectID
dummy_metadata <- system.file("dataset-demo", "dummy_metadata.xlsx", package = "AlpsNMR")
NMRExp_SubjID <- readxl::read_excel(dummy_metadata, sheet = 1)

NMRExp_SubjID
# We can link the SubjectID column of the first excel into the dataset
nmr_dataset <- nmr_meta_add(nmr_dataset, NMRExp_SubjID, by = "NMRExperiment")
nmr_meta_get(nmr_dataset, groups = "external")
# The second excel can use the SubjectID:
SubjID_Age <- readxl::read_excel(dummy_metadata, sheet = 2)
SubjID_Age
# Add the metadata by its SubjectID:
nmr_dataset <- nmr_meta_add(nmr_dataset, SubjID_Age, by = "SubjectID")
# The final loaded metadata:
nmr_meta_get(nmr_dataset, groups = "external")

# Read a tidy excel file:

dataset <- system.file("dataset-demo", package = "AlpsNMR")
nmr_dataset <- nmr_read_samples_dir(dataset)

# At first we just have the NMRExperiment column
nmr_meta_get(nmr_dataset, groups = "external")
# Get a table with NMRExperiment -> SubjectID
dummy_metadata <- system.file("dataset-demo", "dummy_metadata.xlsx", package = "AlpsNMR")

nmr_dataset <- nmr_meta_add_tidy_excel(nmr_dataset, dummy_metadata)
# Updated Metadata:
nmr_meta_get(nmr_dataset, groups = "external")
```

`nmr_meta_export` *Export Metadata to an Excel file*

Description

Export Metadata to an Excel file

Usage

```
nmr_meta_export(
  nmr_dataset,
  xlsx_file,
  groups = c("info", "orig", "title", "external")
)
```

Arguments

`nmr_dataset`	An nmr_dataset_family object
`xlsx_file`	"The .xlsx excel file"
`groups`	A character vector. Use `"external"` for the external metadata or the default for a more generic solution

Value

The Excel file name

See Also

Other metadata functions: `Pipelines`, `nmr_meta_add()`, `nmr_meta_get()`, `nmr_meta_get_column()`, `nmr_meta_groups()`

Other nmr_dataset functions: `nmr_meta_add()`, `nmr_meta_get()`, `nmr_meta_get_column()`

Other nmr_dataset_1D functions: `[.nmr_dataset_1D()`, `format.nmr_dataset_1D()`, `get_integration_with_metadata`
`is.nmr_dataset_1D()`, `nmr_integrate_peak_positions()`, `nmr_integrate_regions()`, `nmr_meta_add()`, `nmr_meta_get()`, `nmr_meta_get_column()`, `nmr_ppm_resolution()`, `print.nmr_dataset_1D()`

Other nmr_dataset_peak_table functions: `nmr_meta_add()`, `nmr_meta_get()`, `nmr_meta_get_column()`

Other import/export functions: `Pipelines`, `files_to_rDolphin()`, `load_and_save_functions`, `nmr_data()`, `nmr_read_bruker_fid()`, `nmr_read_samples()`, `nmr_zip_bruker_samples()`, `save_files_to_rDolphin(`
`save_profiling_output()`, `to_ChemoSpec()`

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
# nmr_meta_export(dataset, "metadata.xlsx")
```

nmr_meta_get *Get metadata*

Description

Get metadata

Usage

```
nmr_meta_get(samples, columns = NULL, groups = NULL)
```

Arguments

samples	a nmr_dataset_family object
columns	Columns to get. By default gets all the columns.
groups	Groups to get. Groups are predefined of columns. Typically "external" for metadata added with nmr_meta_add. Both groups and columns can't be given simultaneously.

Value

a data frame with the injection metadata

See Also

Other metadata functions: Pipelines, nmr_meta_add(), nmr_meta_export(), nmr_meta_get_column(), nmr_meta_groups()

Other nmr_dataset functions: nmr_meta_add(), nmr_meta_export(), nmr_meta_get_column()

Other nmr_dataset_1D functions: [.nmr_dataset_1D(), format.nmr_dataset_1D(), get_integration_with_metadata is.nmr_dataset_1D(), nmr_integrate_peak_positions(), nmr_integrate_regions(), nmr_meta_add(), nmr_meta_export(), nmr_meta_get_column(), nmr_ppm_resolution(), print.nmr_dataset_1D()

Other nmr_dataset_peak_table functions: nmr_meta_add(), nmr_meta_export(), nmr_meta_get_column()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
metadata <- nmr_meta_get(dataset)
```

`nmr_meta_get_column` *Get a single metadata column*

Description

Get a single metadata column

Usage

```
nmr_meta_get_column(samples, column = "NMRExperiment")
```

Arguments

`samples`	a nmr_dataset_family object
`column`	A column to get

Value

A vector with the column

See Also

Other metadata functions: `Pipelines`, `nmr_meta_add()`, `nmr_meta_export()`, `nmr_meta_get()`, `nmr_meta_groups()`

Other nmr_dataset functions: `nmr_meta_add()`, `nmr_meta_export()`, `nmr_meta_get()`

Other nmr_dataset_1D functions: `[.nmr_dataset_1D()`, `format.nmr_dataset_1D()`, `get_integration_with_metadata`
`is.nmr_dataset_1D()`, `nmr_integrate_peak_positions()`, `nmr_integrate_regions()`, `nmr_meta_add()`, `nmr_meta_export()`, `nmr_meta_get()`, `nmr_ppm_resolution()`, `print.nmr_dataset_1D()`

Other nmr_dataset_peak_table functions: `nmr_meta_add()`, `nmr_meta_export()`, `nmr_meta_get()`

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
metadata_column <- nmr_meta_get_column(dataset)
```

nmr_meta_groups *Get the names of metadata groups*

Description

Get the names of metadata groups

Usage

```
nmr_meta_groups(samples)
```

Arguments

samples	a nmr_dataset_family object

Value

A character vector with group names

See Also

Other metadata functions: Pipelines, nmr_meta_add(), nmr_meta_export(), nmr_meta_get(), nmr_meta_get_column()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
metadata_column <- nmr_meta_get_column(dataset)
```

nmr_normalize *Normalize nmr_dataset_1D samples*

Description

The nmr_normalize function is used to normalize all the samples according to a given criteria.

Usage

```
nmr_normalize(
  samples,
  method = c("area", "max", "value", "region", "pqn", "none"),
  ...
)

nmr_normalize_extra_info(samples)
```

Arguments

samples	A nmr_dataset_1D object
method	The criteria to be used for normalization - area: Normalize to the total area - max: Normalize to the maximum intensity - value: Normalize each sample to a user defined value - region: Integrate a region and normalize each sample to that region - pqn: Use Probabalistic Quotient Normalization for normalization - none: Do not normalize at all
...	Method dependent arguments: - method == "value": - value: A numeric vector with the normalization values to use - method == "region": - ppm_range: A chemical shift region to integrate - . . . : Other arguments passed on to nmr_integrate_regions

Details

The aim is to correct from changes between samples, so no matter the criteria used to normalize, once we get the factors (e.g. the areas), we divide them by the median normalization factor to avoid introducing global scaling factors.

The nmr_normalize_extra_info function is used to extract additional information after the normalization. Typically, we want to know what was the actual normalization factor applied to each sample. The extra information includes a plot, representing the dispersion of the normalization factor for each sample.

Value

The nmr_dataset_1D object, with the samples normalized. Further information for diagnostic of the normalization process is also saved and can be extracted by calling nmr_normalize_extra_info() afterwards.

See Also

Other basic functions: nmr_exclude_region()

Examples

```
nmr_dataset <- nmr_dataset_load(system.file("extdata", "nmr_dataset.rds", package = "AlpsNMR"))
nmr_dataset <- nmr_normalize(nmr_dataset, method = "area")
norm_dataset <- nmr_normalize(nmr_dataset)
norm_dataset$plot
nmr_dataset <- nmr_dataset_load(system.file("extdata", "nmr_dataset.rds", package = "AlpsNMR"))
nmr_dataset <- nmr_normalize(nmr_dataset, method = "area")
norm_extra_info <- nmr_normalize_extra_info(nmr_dataset)
norm_extra_info$plot
```

`nmr_pca_build_model` *Build a PCA on for an nmr_dataset*

Description

This function builds a PCA model with all the NMR spectra. Regions with zero values (excluded regions) or near-zero variance regions are automatically excluded from the analysis.

Usage

```
nmr_pca_build_model(
  nmr_dataset,
  ncomp = NULL,
  center = TRUE,
  scale = FALSE,
  ...
)

## S3 method for class 'nmr_dataset_1D'
nmr_pca_build_model(
  nmr_dataset,
  ncomp = NULL,
  center = TRUE,
  scale = FALSE,
  ...
)
```

Arguments

`nmr_dataset`	a nmr_dataset_1D object
`ncomp`	Integer, if data is complete `ncomp` decides the number of components and associated eigenvalues to display from the `pcasvd` algorithm and if the data has missing values, `ncomp` gives the number of components to keep to perform the reconstitution of the data using the NIPALS algorithm. If `NULL`, function sets `ncomp = min(nrow(X),ncol(X))`
`center`	(Default=TRUE) Logical, whether the variables should be shifted to be zero centered. Only set to FALSE if data have already been centered. Alternatively, a vector of length equal the number of columns of `X` can be supplied. The value is passed to `scale`. If the data contain missing values, columns should be centered for reliable results.
`scale`	(Default=FALSE) Logical indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is `FALSE` for consistency with `prcomp` function, but in general scaling is advisable. Alternatively, a vector of length equal the number of columns of `X` can be supplied. The value is passed to `scale`.
`...`	Additional arguments passed on to mixOmics::pca

Value

A PCA model as given by mixOmics::pca with two additional attributes:

- `nmr_data_axis` containing the full ppm axis
- `nmr_included` with the data points included in the model These attributes are used internally by AlpsNMR to create loading plots

See Also

Other PCA related functions: `nmr_pca_outliers()`, `nmr_pca_outliers_filter()`, `nmr_pca_outliers_plot()`, `nmr_pca_outliers_robust()`, `nmr_pca_plots`

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
model <- nmr_pca_build_model(dataset_1D)

dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
model <- nmr_pca_build_model(dataset_1D)
```

`nmr_pca_outliers` *Compute PCA residuals and score distance for each sample*

Description

Compute PCA residuals and score distance for each sample

Usage

```
nmr_pca_outliers(
  nmr_dataset,
  pca_model,
  ncomp = NULL,
  quantile_critical = 0.975
)
```

Arguments

`nmr_dataset`	An nmr_dataset_1D object
`pca_model`	A pca model returned by nmr_pca_build_model
`ncomp`	Number of components to use. Use `NULL` for 90% of the variance
`quantile_critical`	critical quantile

Value

A list with:

- outlier_info: A data frame with the NMRExperiment, the Q residuals and T scores
- ncomp: Number of components used to compute Q and T
- Tscore_critical, QResidual_critical: Critical values, given a quantile, for both Q and T.

See Also

Other PCA related functions: `nmr_pca_build_model()`, `nmr_pca_outliers_filter()`, `nmr_pca_outliers_plot()`, `nmr_pca_outliers_robust()`, `nmr_pca_plots`

Other outlier detection functions: `Pipelines`, `nmr_pca_outliers_filter()`, `nmr_pca_outliers_plot()`, `nmr_pca_outliers_robust()`

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
model <- nmr_pca_build_model(dataset_1D)
outliers_info <- nmr_pca_outliers(dataset_1D, model)
```

---

nmr_pca_outliers_filter

Exclude outliers

Description

Exclude outliers

Usage

```
nmr_pca_outliers_filter(nmr_dataset, pca_outliers)
```

Arguments

nmr_dataset	An nmr_dataset_1D object
pca_outliers	The output from `nmr_pca_outliers()`

Value

An nmr_dataset_1D without the detected outliers

See Also

Other PCA related functions: nmr_pca_build_model(), nmr_pca_outliers(), nmr_pca_outliers_plot(), nmr_pca_outliers_robust(), nmr_pca_plots

Other outlier detection functions: Pipelines, nmr_pca_outliers(), nmr_pca_outliers_plot(), nmr_pca_outliers_robust()

Other subsetting functions: [.nmr_dataset(), [.nmr_dataset_1D(), [.nmr_dataset_peak_table(), filter.nmr_dataset_family()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
model <- nmr_pca_build_model(dataset_1D)
outliers_info <- nmr_pca_outliers(dataset_1D, model)
dataset_whitout_outliers <- nmr_pca_outliers_filter(dataset_1D, outliers_info)
```

---

nmr_pca_outliers_plot *Plot for outlier detection diagnostic*

Description

Plot for outlier detection diagnostic

Usage

```
nmr_pca_outliers_plot(nmr_dataset, pca_outliers, ...)
```

Arguments

nmr_dataset	An nmr_dataset_1D object
pca_outliers	The output from nmr_pca_outliers()
...	Additional parameters passed on to ggplot2::aes() (or now deprecated to ggplot2::aes_string())

Value

A plot for the outlier detection

See Also

Other PCA related functions: nmr_pca_build_model(), nmr_pca_outliers(), nmr_pca_outliers_filter(), nmr_pca_outliers_robust(), nmr_pca_plots

Other outlier detection functions: Pipelines, nmr_pca_outliers(), nmr_pca_outliers_filter(), nmr_pca_outliers_robust()

Examples

```
# dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
# dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
# dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
# model <- nmr_pca_build_model(dataset_1D)
# outliers_info <- nmr_pca_outliers(dataset_1D, model)
# nmr_pca_outliers_plot(dataset_1D, outliers_info)
```

`nmr_pca_outliers_robust` *Outlier detection through robust PCA*

Description

Outlier detection through robust PCA

Usage

```
nmr_pca_outliers_robust(nmr_dataset, ncomp = 5)
```

Arguments

`nmr_dataset`	An nmr_dataset_1D object
`ncomp`	Number of rPCA components to use We have observed that the statistical test used as a threshold for outlier detection usually flags as outliers too many samples, due possibly to a lack of gaussianity As a workaround, a heuristic method has been implemented: We know that in the Q residuals vs T scores plot from `nmr_pca_outliers_plot()` outliers are on the right or on the top of the plot, and quite separated from non-outlier samples. To determine the critical value, both for Q and T, we find the biggest gap between samples in the plot and use as critical value the center of the gap. This approach seems to work well when there are outliers, but it fails when there isn't any outlier. For that case, the gap would be placed anywhere and that is not desirable as many samples would be incorrectly flagged. The second assumption that we use is that no more than 10\ the samples may pass our critical value. If more than 10\ pass the critical value, then we assume that our heuristics are not reasonable and we don't set any critical limit.

Value

A list similar to nmr_pca_outliers

See Also

Other PCA related functions: nmr_pca_build_model(), nmr_pca_outliers(), nmr_pca_outliers_filter(), nmr_pca_outliers_plot(), nmr_pca_plots

Other outlier detection functions: Pipelines, nmr_pca_outliers(), nmr_pca_outliers_filter(), nmr_pca_outliers_plot()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
outliers_info <- nmr_pca_outliers_robust(dataset_1D)
```

---

nmr_pca_plots *Plotting functions for PCA*

Description

Plotting functions for PCA

Usage

```
nmr_pca_plot_variance(pca_model)

nmr_pca_scoreplot(nmr_dataset, pca_model, comp = seq_len(2), ...)

nmr_pca_loadingplot(pca_model, comp)
```

Arguments

pca_model	A PCA model trained with nmr_pca_build_model
nmr_dataset	an nmr_dataset_1D object
comp	Components to represent
...	Additional aesthetics passed on to ggplot2::aes (use bare unquoted names)

Value

Plot of PCA

See Also

Other PCA related functions: nmr_pca_build_model(), nmr_pca_outliers(), nmr_pca_outliers_filter(), nmr_pca_outliers_plot(), nmr_pca_outliers_robust()

Examples

```
dataset_1D <- nmr_dataset_load(system.file("extdata", "nmr_dataset.rds", package = "AlpsNMR"))
model <- nmr_pca_build_model(dataset_1D)
nmr_pca_plot_variance(model)
nmr_pca_scoreplot(dataset_1D, model)
nmr_pca_loadingplot(model, 1)
```

---

nmr_peak_clustering *Peak clustering*

Description

Peak clustering

Usage

```
nmr_peak_clustering(
  peak_data,
  peak2peak_dist = NULL,
  num_clusters = NULL,
  max_dist_thresh_ppb = NULL,
  verbose = FALSE
)
```

Arguments

peak_data	The peak list
peak2peak_dist	The distances obtained with nmr_get_peak_distances. If NULL it is computed from peak_data
num_clusters	If you want to fix the number of clusters. Leave NULL if you want to estimate it
max_dist_thresh_ppb	To estimate the number of clusters, we enforce a limit on how far two peaks of the same cluster may be. By default this threshold will be computed as 3 times the median peak width (gamma), as given in the peak list.
verbose	A logical vector to print additional information

Value

A list including:

- The peak_data with an additional "cluster" column
- cluster: the hierarchical cluster
- num_clusters: an estimation of the number of clusters
- num_cluster_estimation: A list with tables and plots to justify the number of cluster estimation

Examples

```
peak_data <- data.frame(
    NMRExperiment = c("10", "10", "20", "20"),
    peak_id = paste0("Peak", 1:4),
    ppm = c(1, 2, 1.1, 2.2),
    gamma_ppb = 100
)
clustering_result <- nmr_peak_clustering(peak_data)
peak_data <- clustering_result$peak_data
stopifnot("cluster" %in% colnames(peak_data))
```

---

nmr_peak_clustering_plot

Plot clustering results

Description

Plot clustering results

Usage

```
nmr_peak_clustering_plot(
  dataset,
  peak_list_clustered,
  NMRExperiments,
  chemshift_range,
  baselineThresh = NULL
)
```

Arguments

`dataset`	The nmr_dataset_1D object
`peak_list_clustered`	A peak list table with a clustered column
`NMRExperiments`	Two and only two experiments to compare in the plot
`chemshift_range`	A region, make it so it does not cover a huge range (maybe 1ppm or less)
`baselineThresh`	If given (as returned from the `nmr_baseline_threshold()`) the baseline threshold will be plotted. This can be useful to diagnose whether a peak is missing due to this threshold or due to other parameters (e.g. `SNR.Th`). See `nmr_detect_peaks()`.

Value

A plot of the two experiments in the given chemshift range, with lines connecting peaks identified as the same and dots showing peaks without pairs

`nmr_ppm_resolution` *PPM resolution of the spectra*

Description

The function gets the ppm resolution of the dataset using the median of the difference of data points.

Usage

```
nmr_ppm_resolution(nmr_dataset)

## S3 method for class 'nmr_dataset'
nmr_ppm_resolution(nmr_dataset)

## S3 method for class 'nmr_dataset_1D'
nmr_ppm_resolution(nmr_dataset)
```

Arguments

`nmr_dataset`	An object containing NMR samples

Value

Numeric (the ppm resolution, measured in ppms)

See Also

Other nmr_dataset_1D functions: `[.nmr_dataset_1D()`, `format.nmr_dataset_1D()`, `get_integration_with_metadata`
`is.nmr_dataset_1D()`, `nmr_integrate_peak_positions()`, `nmr_integrate_regions()`, `nmr_meta_add()`,
`nmr_meta_export()`, `nmr_meta_get()`, `nmr_meta_get_column()`, `print.nmr_dataset_1D()`

Examples

```
nmr_dataset <- nmr_dataset_load(system.file("extdata", "nmr_dataset.rds", package = "AlpsNMR"))
nmr_ppm_resolution(nmr_dataset)
message("the ppm resolution of this dataset is ", nmr_ppm_resolution(nmr_dataset), " ppm")

nmr_dataset <- nmr_dataset_load(system.file("extdata", "nmr_dataset.rds", package = "AlpsNMR"))
nmr_ppm_resolution(nmr_dataset)
message("the ppm resolution of this dataset is ", nmr_ppm_resolution(nmr_dataset), " ppm")

nmr_dataset <- nmr_dataset_load(system.file("extdata", "nmr_dataset.rds", package = "AlpsNMR"))
nmr_ppm_resolution(nmr_dataset)
message("the ppm resolution of this dataset is ", nmr_ppm_resolution(nmr_dataset), " ppm")
```

nmr_read_bruker_fid *Read Free Induction Decay file*

Description

Reads an FID file. This is a very simple function.

Usage

```
nmr_read_bruker_fid(sample_name, endian = "little")
```

Arguments

sample_name	A single sample name
endian	Endianness of the fid file ("little" by default, use "big" if acqus$BYTORDA == 1)

Value

A numeric vector with the free induction decay values

See Also

Other import/export functions: Pipelines, files_to_rDolphin(), load_and_save_functions, nmr_data(), nmr_meta_export(), nmr_read_samples(), nmr_zip_bruker_samples(), save_files_to_rDolphin(), save_profiling_output(), to_ChemoSpec()

Examples

```
fid <- nmr_read_bruker_fid("sample.fid")
```

nmr_read_samples *Read NMR samples*

Description

These functions load samples from files and return a nmr_dataset.

Usage

```
nmr_read_samples_dir(
  samples_dir,
  format = "bruker",
  pulse_sequence = NULL,
  metadata_only = FALSE,
  ...
)

nmr_read_samples(
  sample_names,
  format = "bruker",
  pulse_sequence = NULL,
  metadata_only = FALSE,
  ...
)
```

Arguments

samples_dir	A directory or directories that contain multiple samples
format	Either "bruker" or "jdx"
pulse_sequence	If it is set to a pulse sequence ("NOESY", "JRES", "CPMG"...) it will only load the samples that match that pulse sequence.
metadata_only	A logical, to load only metadata (default: FALSE)
...	Arguments passed on to read_bruker_pdata pdata_file File name of the binary NMR data to load. Usually "1r". If NULL, it is autodetected based on the dimension sample_path A character path of the sample directory pdata_path Path from sample_path to the preprocessed data all_components If FALSE load only the real component. Otherwise load the real and imaginary components read_pdata_title If TRUE also reads metadata from pdata title file.
sample_names	A character vector with file or directory names.

Value

a nmr_dataset object

See Also

read_bruker_pdata()

Other import/export functions: Pipelines, files_to_rDolphin(), load_and_save_functions, nmr_data(), nmr_meta_export(), nmr_read_bruker_fid(), nmr_zip_bruker_samples(), save_files_to_rDolphin() save_profiling_output(), to_ChemoSpec()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)

dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
zip_files <- fs::dir_ls(dir_to_demo_dataset, glob = "*.zip")
dataset <- nmr_read_samples(sample_names = zip_files)
```

`nmr_zip_bruker_samples` *Create one zip file for each brucker sample path*

Description

Create one zip file for each brucker sample path

Usage

```
nmr_zip_bruker_samples(path, workdir, overwrite = FALSE, ...)
```

Arguments

`path`	Character vector with sample directories
`workdir`	Directory to store zip files
`overwrite`	Should existing zip files be overwritten?
`...`	Passed to utils::zip

Value

A character vector of the same length as path, with the zip file names

See Also

Other import/export functions: `Pipelines`, `files_to_rDolphin()`, `load_and_save_functions`, `nmr_data()`, `nmr_meta_export()`, `nmr_read_bruker_fid()`, `nmr_read_samples()`, `save_files_to_rDolphin()`, `save_profiling_output()`, `to_ChemoSpec()`

Examples

```
save_zip_files_to <- tempfile(pattern = "zip_file_storage_")
where_your_samples_are <- tempfile(pattern = "where_your_samples_are")
# prepare sample:
zip::unzip(
  system.file("dataset-demo", "10.zip", package = "AlpsNMR"),
  exdir = where_your_samples_are
)
```

```
outpaths <- nmr_zip_bruker_samples(
    list.files(where_your_samples_are, full.names = TRUE),
    workdir = save_zip_files_to
)
```

Parameters_blood *to rDolphin*

Description

Parameters for blood (plasma/serum) samples profiling

Details

The template Parameters_blood contains the chosen normalization approach (by default, PQN), the Spectometer Frequency (by default, 600.04MHz), alignment (by default, TSP 0.00 ppm), bucket resolution (by default, 0.00023)

References

github.com/danielcanueto/rDolphin

Examples

```
data("Parameters_blood")
Parameters_blood
```

Parameters_cell *Parameters for cell samples profiling*

Description

The template Parameters_cell contains the chosen normalization approach (by default, PQN), the Spectometer Frequency (by default, 600.04MHz), alignment (by default, TSP 0.00 ppm), bucket resolution (by default, 0.00023)

References

github.com/danielcanueto/rDolphin

Examples

```
data("Parameters_cell")
Parameters_cell
```

Parameters_urine *Parameters for urine samples profiling*

Description

The template Parameters_urine contains the chosen normalization approach (by default, PQN), the Spectometer Frequency (by default, 600.04MHz), alignment (by default, TSP 0.00 ppm), bucket resolution (by default, 0.00023)

References

github.com/danielcanueto/rDolphin

Examples

```
data("Parameters_urine")
Parameters_urine
```

peaklist_accept_peaks *Peak list: Create an* accepted *column based on some criteria*

Description

Peak list: Create an accepted column based on some criteria

Usage

```
peaklist_accept_peaks(
  peak_data,
  nmr_dataset,
  nrmse_max = Inf,
  area_min = 0,
  area_max = Inf,
  ppm_min = -Inf,
  ppm_max = Inf,
  keep_rejected = TRUE,
  verbose = FALSE
)
```

Arguments

peak_data	The peak list (a data frame)
nmr_dataset	The nmr_dataset where the peak_data was computed from
nrmse_max	The normalized root mean squared error of the lorentzian peak fitting must be less than or equal to this value

area_min	Peak areas must be larger or equal to this value
area_max	Peak areas must be smaller or equal to this value
ppm_min	The peak apex must be above this value
ppm_max	The peak apex must be below this value
keep_rejected	If FALSE, removes those peaks that do not satisfy the criteria and remove the accepted column (since all would be accepted)
verbose	Print informational message

Value

The peak_data, with a new accepted column (or maybe some filtered rows)

Examples

```
# Fake data:
nmr_dataset <- new_nmr_dataset_1D(
    1:10,
    matrix(c(1:5, 4:2, 3, 0), nrow = 1),
    list(external = data.frame(NMRExperiment = "10"))
)
peak_data <- data.frame(
    peak_id = c("Peak1", "Peak2"),
    NMRExperiment = c("10", "10"),
    ppm = c(5, 9),
    pos = c(5, 9),
    intensity = c(5, 3),
    ppm_infl_min = c(3, 8),
    ppm_infl_max = c(7, 10),
    gamma_ppb = c(1, 1),
    area = c(25, 3),
    norm_rmse = c(0.01, 0.8)
)
# Create the accepted column:
peak_data <- peaklist_accept_peaks(peak_data, nmr_dataset, area_min = 10, keep_rejected = FALSE)
stopifnot(identical(peak_data$peak_id, "Peak1"))
```

---

peaklist_fit_lorentzians

Fit lorentzians to each peak to estimate areas

Description

The different methods are available for benchmarking while developing, we should pick one.

Usage

```
peaklist_fit_lorentzians(
  peak_data,
  nmr_dataset,
 amplitude_method = c("intensity", "2nd_derivative", "intensity_without_baseline"),
  refine_peak_model = c("none", "peak", "2nd_derivative")
)
```

Arguments

peak_data	The peak data
nmr_dataset	The nmr_dataset object with the data. This function for now assumes nmr_dataset is NOT be baseline corrected
amplitude_method	The method to estimate the amplitude. It may be: • "intensity". The amplitude of the peak is proportional to the raw intensity at the apex. This is a bad estimation if the intensity includes a baseline, because the amplitude of the peak will be overestimated • "2nd_derivative": The amplitude of the peak is proportional to the second derivative of the raw intensity signal at the apex. This method aims to correct the "intensity" method, since it is expected that the baseline will be mostly removed when considering the 2nd derivative of the spectrum. The 2nd derivative is calculated with a 2nd order Savitzky-Golay filter of 21 points. • "intensity_without_baseline": A baseline is estimated on the whole spectra and subtracted from it. Then the peak amplitude is proportional to the corrected intensity at the apex (as in the "intensity" method).
refine_peak_model	Whether a non linear least squares fitting should be used to refine the estimated parameters. It can be: • "none": Do not refine using nls. • "peak": Use a lorentzian peak model and the baseline corrected spectra. • "2nd_derivative":

Details

- gamma is estimated using the inflection points of the signal and fitting them to the lorentzian inflection points
- A is estimated using the amplitude_method below
- The peak position (x_0) is given in peak_data

Those estimations may be refined with non-linear least squares using refine_peak_model. If the nls does not converge, the initial estimations are kept. Convergence -and other nls errors- are saved for further reference and diagnostic. Use attr(peak_data_fitted, "errors") to retreive the error messages, where peak_data_fitted is assumed to be the output of this function. The refining improves gamma, A and x_0.

The baseline estimation (when calculated, see the arguments) is set to Asymmetric Least Squares with lambda = 6, p=0.05, maxit=20 and it is probably not optimal... yet.

Value

The given data frame `peak_data`, with added columns:

- inflection points,
- gamma
- area
- a norm_rmse fitting error

As well as some attributes

- "errors": A data frame with any error in the peak fitting
- "fit_baseline": Whether the method used has any consideration for the baseline of the signal (maybe not very useful attribute)
- "method_description": A textual description of what we did, to include it in plots

---

`Peak_detection` *Peak detection for NMR*

Description

Peak detection for NMR

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
nmr_dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
# Low resolution:
dataset_1D <- nmr_interpolate_1D(nmr_dataset, axis = c(min = -0.5, max = 10, by = 0.001))
dataset_1D <- nmr_exclude_region(dataset_1D, exclude = list(water = c(4.7, 5)))

# 1. Optimize peak detection parameters:
range_without_peaks <- c(9.5, 10)
# Choose a region without peaks:
plot(dataset_1D, chemshift_range = range_without_peaks)
baselineThresh <- nmr_baseline_threshold(dataset_1D, range_without_peaks = range_without_peaks)
# Plot to check the baseline estimations
nmr_baseline_threshold_plot(
    dataset_1D,
    baselineThresh,
    NMRExperiment = "all",
    chemshift_range = range_without_peaks
)

# 1.Peak detection in the dataset.
peak_data <- nmr_detect_peaks(
    dataset_1D,
    nDivRange_ppm = 0.1, # Size of detection segments
    scales = seq(1, 16, 2),
```

```
    baselineThresh = NULL, # Minimum peak intensity
    SNR.Th = 4, # Signal to noise ratio
    range_without_peaks = range_without_peaks, # To estimate
)

sample_10 <- filter(dataset_1D, NMRExperiment == "10")
# nmr_detect_peaks_plot(sample_10, peak_data, "NMRExp_ref")

peaks_detected <- nmr_detect_peaks_tune_snr(
    sample_10,
    SNR_thresholds = seq(from = 2, to = 3, by = 0.5),
    nDivRange_ppm = 0.03,
    scales = seq(1, 16, 2),
    baselineThresh = 0
)


# 2.Find the reference spectrum to align with.
NMRExp_ref <- nmr_align_find_ref(dataset_1D, peak_data)

# 3.Spectra alignment using the ref spectrum and a maximum alignment shift
nmr_dataset <- nmr_align(dataset_1D, # the dataset
    peak_data, # detected peaks
    NMRExp_ref = NMRExp_ref, # ref spectrum
    maxShift_ppm = 0.0015, # max alignment shift
    acceptLostPeak = FALSE
) # lost peaks

# 4.PEAK INTEGRATION (please, consider previous normalization step).
# First we take the peak table from the reference spectrum
peak_data_ref <- filter(peak_data, NMRExperiment == NMRExp_ref)

# Then we integrate spectra considering the peaks from the ref spectrum
nmr_peak_table <- nmr_integrate_peak_positions(
    samples = nmr_dataset,
    peak_pos_ppm = peak_data_ref$ppm,
    peak_width_ppm = NULL
)

validate_nmr_dataset_peak_table(nmr_peak_table)

# If you wanted the final peak table before machine learning you can run
nmr_peak_table_completed <- get_integration_with_metadata(nmr_peak_table)
```

`permutation_test_model` *Permutation test*

Description

Make permutations with data and default settings from an nmr_data_analysis_method

Usage

```
permutation_test_model(
  dataset,
  y_column,
  identity_column,
  external_val,
  internal_val,
  data_analysis_method,
  nPerm = 50
)
```

Arguments

dataset	An nmr_dataset_family object
y_column	A string with the name of the y column (present in the metadata of the dataset)
identity_column	NULL or a string with the name of the identity column (present in the metadata of the dataset).
external_val, internal_val	A list with two elements: iterations and test_size. See random_subsampling for further details
data_analysis_method	An nmr_data_analysis_method object
nPerm	number of permutations

Value

A permutation matrix with permuted values

Examples

```
# Data analysis for a table of integrated peaks

## Generate an artificial nmr_dataset_peak_table:
### Generate artificial metadata:
num_samples <- 32 # use an even number in this example
num_peaks <- 20
metadata <- data.frame(
    NMRExperiment = as.character(1:num_samples),
    Condition = rep(c("A", "B"), times = num_samples / 2)
)

### The matrix with peaks
peak_means <- runif(n = num_peaks, min = 300, max = 600)
peak_sd <- runif(n = num_peaks, min = 30, max = 60)
peak_matrix <- mapply(function(mu, sd) rnorm(num_samples, mu, sd),
    mu = peak_means, sd = peak_sd
)
colnames(peak_matrix) <- paste0("Peak", 1:num_peaks)
```

```
## Artificial differences depending on the condition:
peak_matrix[metadata$Condition == "A", "Peak2"] <-
    peak_matrix[metadata$Condition == "A", "Peak2"] + 70

peak_matrix[metadata$Condition == "A", "Peak6"] <-
    peak_matrix[metadata$Condition == "A", "Peak6"] - 60

### The nmr_dataset_peak_table
peak_table <- new_nmr_dataset_peak_table(
    peak_table = peak_matrix,
    metadata = list(external = metadata)
)

methodology <- plsda_auroc_vip_method(ncomp = 3)
model <- nmr_data_analysis(
    peak_table,
    y_column = "Condition",
    identity_column = NULL,
    external_val = list(iterations = 3, test_size = 0.25),
    internal_val = list(iterations = 3, test_size = 0.25),
    data_analysis_method = methodology
)

p <- permutation_test_model(peak_table,
    y_column = "Condition",
    identity_column = NULL,
    external_val = list(iterations = 3, test_size = 0.25),
    internal_val = list(iterations = 3, test_size = 0.25),
    data_analysis_method = methodology,
    nPerm = 10
)
```

`permutation_test_plot` *Permutation test plot*

Description

Plot permutation test using actual model and permutated models

Usage

```
permutation_test_plot(
  nmr_data_analysis_model,
  permMatrix,
  xlab = "AUCs",
  xlim,
  ylim = NULL,
  breaks = "Sturges",
```

```
  main = "Permutation test"
)
```

Arguments

nmr_data_analysis_model	A nmr_data_analysis_model
permMatrix	A permutation fitness outcome from permutation_test_model
xlab	optional xlabel
xlim	optional x-range
ylim	otional y-range
breaks	optional custom histogram breaks (defaults to 'sturges')
main	optional plot title (or TRUE for autoname)

Value

A plot with the comparison between the actual model versus the permuted models

Examples

```
# Data analysis for a table of integrated peaks

## Generate an artificial nmr_dataset_peak_table:
### Generate artificial metadata:
num_samples <- 32 # use an even number in this example
num_peaks <- 20
metadata <- data.frame(
    NMRExperiment = as.character(1:num_samples),
    Condition = rep(c("A", "B"), times = num_samples / 2)
)

### The matrix with peaks
peak_means <- runif(n = num_peaks, min = 300, max = 600)
peak_sd <- runif(n = num_peaks, min = 30, max = 60)
peak_matrix <- mapply(function(mu, sd) rnorm(num_samples, mu, sd),
    mu = peak_means, sd = peak_sd
)
colnames(peak_matrix) <- paste0("Peak", 1:num_peaks)

## Artificial differences depending on the condition:
peak_matrix[metadata$Condition == "A", "Peak2"] <-
    peak_matrix[metadata$Condition == "A", "Peak2"] + 70

peak_matrix[metadata$Condition == "A", "Peak6"] <-
    peak_matrix[metadata$Condition == "A", "Peak6"] - 60

### The nmr_dataset_peak_table
peak_table <- new_nmr_dataset_peak_table(
    peak_table = peak_matrix,
    metadata = list(external = metadata)
```

```
)

methodology <- plsda_auroc_vip_method(ncomp = 3)
model <- nmr_data_analysis(
    peak_table,
    y_column = "Condition",
    identity_column = NULL,
    external_val = list(iterations = 3, test_size = 0.25),
    internal_val = list(iterations = 3, test_size = 0.25),
    data_analysis_method = methodology
)

p <- permutation_test_model(peak_table,
    y_column = "Condition",
    identity_column = NULL,
    external_val = list(iterations = 3, test_size = 0.25),
    internal_val = list(iterations = 3, test_size = 0.25),
    data_analysis_method = methodology,
    nPerm = 10
)

permutation_test_plot(model, p)
```

Pipelines *Pipelines*

Description

Uses nmr_pca_outliers_robust to perform the detection of outliers

Normalize the full spectra to the internal calibrant region, then exclude that region and finally perform PQN normalization.

Usage

```
pipe_load_samples(samples_dir, glob = "*0", output_dir = NULL)

pipe_add_metadata(nmr_dataset_rds, excel_file, output_dir)

pipe_interpolate_1D(nmr_dataset_rds, axis, output_dir)

pipe_exclude_regions(nmr_dataset_rds, exclude, output_dir)

pipe_outlier_detection(nmr_dataset_rds, output_dir)

pipe_filter_samples(nmr_dataset_rds, conditions, output_dir)

pipe_peakdet_align(
```

```
  nmr_dataset_rds,
  nDivRange_ppm = 0.1,
  scales = seq(1, 16, 2),
  baselineThresh = 0.01,
  SNR.Th = -1,
  maxShift_ppm = 0.0015,
  acceptLostPeak = FALSE,
  output_dir = NULL
)

pipe_peak_integration(
  nmr_dataset_rds,
  peak_det_align_dir,
  peak_width_ppm,
  output_dir
)

pipe_normalization(
  nmr_dataset_rds,
  internal_calibrant = NULL,
  output_dir = NULL
)
```

Arguments

samples_dir	The directory where the samples are
glob	A wildcard aka globbing pattern (e.g. `*.csv`) passed on to `grep()` to filter paths.
output_dir	The output directory for this pipe element
nmr_dataset_rds	The nmr_dataset.rds file name coming from previous nodes
excel_file	An excel file name. See details for the requirements The excel file can have one or more sheets. The excel sheets need to be as simple as possible: One header column on the first row and values below. Each of the sheets contain metadata that has to be integrated. The merge (technically a left join) is done using the first column of each sheet as key. In practical terms this means that the first sheet of the excel file MUST start with an "NMRExperiment" column, and as many additional columns to add (e.g. FluidXBarcode, SampleCollectionDate, TimePoint and SubjectID). The second sheet can have as the first column any of the already added columns, for instance the "SubjectID", and any additional columns (e.g. Gender, Age). The first column on each sheet, named the key column, MUST have unique values. For instance, a sheet starting with "SubjectID" MUST specify each subject ID only once (without repetitions).
axis	The ppm axis range and optionally the ppm step. Set it to `NULL` for autodetection
exclude	A list with regions to be removed Typically: `exclude = list(water = c(4.7, 5.0))`

`conditions` A character vector with conditions to filter metadata. The `conditions` parameter should be a character vector of valid R logical conditions. Some examples:

- conditions <- 'Gender == "Female"'
- conditions <- 'Cohort == "Chuv"'
- conditions <- 'TimePoint %in% c("T0", "T31")'
- conditions <- c(Cohort == "Chuv", 'TimePoint %in% c("T0", "T31")')

Only samples fullfilling all the given conditions are kept in further analysis.

`nDivRange_ppm` Segment size, in ppms, to divide the spectra and search for peaks.

`scales` The parameter of peakDetectionCWT function of MassSpecWavelet package, look it up in the original function.

`baselineThresh` All peaks with intensities below the thresholds are excluded. Either:

- A numeric vector of length the number of samples. Each number is a threshold for that sample
- A single number. All samples use this number as baseline threshold.
- `NULL`. If that's the case, a default function is used (`nmr_baseline_threshold()`), which assumes that there is no signal in the region 9.5-10 ppm.

`SNR.Th` The parameter of peakDetectionCWT function of MassSpecWavelet package, look it up in the original function. If you set -1, the function will itself recompute this value.

`maxShift_ppm` The maximum shift allowed, in ppm

`acceptLostPeak` This is an option for users, TRUE is the default value. If the users believe that all the peaks in the peak list are true positive, change it to FALSE.

`peak_det_align_dir`
Output directory from pipe_peakdet_align

`peak_width_ppm` A peak width in ppm

`internal_calibrant`
A ppm range where the internal calibrant is, or `NULL`.

Details

If there is no internal calibrant, only the PQN normalization is done.

Value

This function saves the result to the output directory

This function saves the result to the output directory

This function saves the result to the output directory

This function saves the result to the output directory

This function saves the result to the output directory

Pipeline: Filter samples according to metadata conditions

Pipeline: Peak detection and Alignment

Pipeline: Peak integration

Pipe: Full spectra normalization

See Also

Other import/export functions: files_to_rDolphin(), load_and_save_functions, nmr_data(), nmr_meta_export(), nmr_read_bruker_fid(), nmr_read_samples(), nmr_zip_bruker_samples(), save_files_to_rDolphin(), save_profiling_output(), to_ChemoSpec()

Other metadata functions: nmr_meta_add(), nmr_meta_export(), nmr_meta_get(), nmr_meta_get_column(), nmr_meta_groups()

Other outlier detection functions: nmr_pca_outliers(), nmr_pca_outliers_filter(), nmr_pca_outliers_plot(), nmr_pca_outliers_robust()

Other peak detection functions: nmr_baseline_threshold(), nmr_detect_peaks(), nmr_detect_peaks_plot(), nmr_detect_peaks_plot_overview(), nmr_detect_peaks_tune_snr(), nmr_identify_regions_blood(), nmr_identify_regions_cell(), nmr_identify_regions_urine(), nmr_integrate_regions()

Other alignment functions: nmr_align(), nmr_align_find_ref()

Other peak integration functions: get_integration_with_metadata(), nmr_identify_regions_blood(), nmr_identify_regions_cell(), nmr_identify_regions_urine(), nmr_integrate_peak_positions(), nmr_integrate_regions()

Examples

```
## Example of pipeline usage
## There are differet ways of load the dataset
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
# excel_file <- system.file("dataset-demo",
#                           "dummy_metadata.xlsx",
#                           package = "AlpsNMR")
# output_dir <- tempdir()

## Load samples with pipes
# pipe_load_samples(dir_to_demo_dataset,
#                   glob = "*.zip",
#                   output_dir = "../pipe_output")

## Another way to load it
# nmr_dataset <- nmr_read_samples_dir(dir_to_demo_dataset)

## Saving the dataset in a .rds file
# nmr_dataset_rds <- tempfile(fileext = ".rds")
# nmr_dataset_save(nmr_dataset, nmr_dataset_rds)

## Interpolation
# pipe_interpolate_1D(nmr_dataset_rds,
#                     axis = c(min = -0.5, max = 10, by = 2.3E-4),
#                     output_dir)

## Get the new path, based in output_dir
# nmr_dataset_rds <- paste(output_dir, "\", "nmr_dataset.rds", sep = "", collapse = NULL)

## Adding metadata to samples
# pipe_add_metadata(nmr_dataset_rds = nmr_dataset_rds, output_dir = output_dir,
#                   excel_file = excel_file)
```

```
## Filtering samples
# conditions <- 'SubjectID == "Ana"'
# pipe_filter_samples(nmr_dataset_rds, conditions, output_dir)

## Outlier detection
# pipe_outlier_detection(nmr_dataset_rds, output_dir)

## Exclude regions
# exclude_regions <- list(water = c(5.1, 4.5))
# pipe_exclude_regions(nmr_dataset_rds, exclude_regions, output_dir)

## peak aling
# pipe_peakdet_align(nmr_dataset_rds, output_dir = output_dir)

## peak integration
# pipe_peak_integration(nmr_dataset_rds,
#                       peak_det_align_dir = output_dir,
#                       peak_width_ppm = 0.006, output_dir)

## Normalization
# pipe_normalization(nmr_dataset_rds, output_dir = output_dir)
```

`plot.nmr_dataset_1D` *Plot an nmr_dataset_1D*

Description

Plot an nmr_dataset_1D

Usage

```
## S3 method for class 'nmr_dataset_1D'
plot(
  x,
  NMRExperiment = NULL,
  chemshift_range = NULL,
  interactive = FALSE,
  quantile_plot = NULL,
  quantile_colors = NULL,
  ...
)
```

Arguments

`x`	a nmr_dataset_1D object
`NMRExperiment`	A character vector with the NMRExperiments to include. Use "all" to include all experiments.

chemshift_range
: range of the chemical shifts to be included. Can be of length 3 to include the resolution in the third element (e.g. `c(0.2, 0.8, 0.005)`)

interactive
: if `TRUE` return an interactive plotly plot, otherwise return a ggplot one.

quantile_plot
: If `TRUE` plot the 10\ If two numbers between 0 and 1 are given then a custom percentile can be plotted

quantile_colors
: A vector with the colors for each of the quantiles

...
: arguments passed to ggplot2::aes (or to ggplot2::aes_string, being deprecated).

Value

The plot

See Also

Other plotting functions: `plot_interactive()`

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
# dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
# dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
# plot(dataset_1D)
```

---

plot_bootstrap_multimodel

Bootstrap plot predictions

Description

Bootstrap plot predictions

Usage

```
plot_bootstrap_multimodel(bp_results, dataset, y_column, plot = TRUE)
```

Arguments

bp_results
: bp_kfold_VIP_analysis results

dataset
: An nmr_dataset_family object

y_column
: A string with the name of the y column (present in the metadata of the dataset)

plot
: A boolean that indicate if results are plotted or not

Value

A plot of the results or a ggplot object

Examples

```
# Data analysis for a table of integrated peaks

## Generate an artificial nmr_dataset_peak_table:
### Generate artificial metadata:
num_samples <- 64 # use an even number in this example
num_peaks <- 20
metadata <- data.frame(
    NMRExperiment = as.character(1:num_samples),
    Condition = rep(c("A", "B"), times = num_samples / 2)
)

### The matrix with peaks
peak_means <- runif(n = num_peaks, min = 300, max = 600)
peak_sd <- runif(n = num_peaks, min = 30, max = 60)
peak_matrix <- mapply(function(mu, sd) rnorm(num_samples, mu, sd),
    mu = peak_means, sd = peak_sd
)
colnames(peak_matrix) <- paste0("Peak", 1:num_peaks)

## Artificial differences depending on the condition:
peak_matrix[metadata$Condition == "A", "Peak2"] <-
    peak_matrix[metadata$Condition == "A", "Peak2"] + 70

peak_matrix[metadata$Condition == "A", "Peak6"] <-
    peak_matrix[metadata$Condition == "A", "Peak6"] - 60

### The nmr_dataset_peak_table
peak_table <- new_nmr_dataset_peak_table(
    peak_table = peak_matrix,
    metadata = list(external = metadata)
)

## We will use bootstrap and permutation method for VIPs selection
## in a a k-fold cross validation
# bp_results <- bp_kfold_VIP_analysis(peak_table, # Data to be analized
#                             y_column = "Condition", # Label
#                             k = 3,
#                             nbootstrap = 10)

# message("Selected VIPs are: ", bp_results$importarn_vips)

# plot_bootstrap_multimodel(bp_results, peak_table, "Condition")
```

plot_interactive *Plots in WebGL*

Description

Plots in WebGL

Usage

```
plot_interactive(plt, html_filename, overwrite = NULL)
```

Arguments

plt	A plot created with plotly or ggplot2
html_filename	The file name where the plot will be saved
overwrite	Overwrite the lib/ directory (use NULL to prompt the user)

Value

The html_filename

See Also

Other plotting functions: plot.nmr_dataset_1D()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
# plot <- plot(dataset_1D)
# html_plot_interactive <- plot_interactive(plot, "html_plot_interactive.html")
```

plot_plsda_multimodel *Multi PLDSA model plot predictions*

Description

Multi PLDSA model plot predictions

Usage

```
plot_plsda_multimodel(model, plot = TRUE)
```

Arguments

model	A nmr_data_analysis_model
plot	A boolean that indicate if results are plotted or not

Value

A plot of the results or a ggplot object

Examples

```
#' # Data analysis for a table of integrated peaks

## Generate an artificial nmr_dataset_peak_table:
### Generate artificial metadata:
num_samples <- 32 # use an even number in this example
num_peaks <- 20
metadata <- data.frame(
    NMRExperiment = as.character(1:num_samples),
    Condition = rep(c("A", "B"), times = num_samples / 2)
)

### The matrix with peaks
peak_means <- runif(n = num_peaks, min = 300, max = 600)
peak_sd <- runif(n = num_peaks, min = 30, max = 60)
peak_matrix <- mapply(function(mu, sd) rnorm(num_samples, mu, sd),
    mu = peak_means, sd = peak_sd
)
colnames(peak_matrix) <- paste0("Peak", 1:num_peaks)

## Artificial differences depending on the condition:
peak_matrix[metadata$Condition == "A", "Peak2"] <-
    peak_matrix[metadata$Condition == "A", "Peak2"] + 70

peak_matrix[metadata$Condition == "A", "Peak6"] <-
    peak_matrix[metadata$Condition == "A", "Peak6"] - 60

### The nmr_dataset_peak_table
peak_table <- new_nmr_dataset_peak_table(
    peak_table = peak_matrix,
    metadata = list(external = metadata)
)

## We will use a double cross validation, splitting the samples with random
## subsampling both in the external and internal validation.
## The classification model will be a PLSDA, exploring at maximum 3 latent
## variables.
## The best model will be selected based on the area under the ROC curve
methodology <- plsda_auroc_vip_method(ncomp = 1)
model <- nmr_data_analysis(
    peak_table,
    y_column = "Condition",
    identity_column = NULL,
```

```
    external_val = list(iterations = 2, test_size = 0.25),
    internal_val = list(iterations = 2, test_size = 0.25),
    data_analysis_method = methodology
)

# plot_plsda_multimodel(model)
```

`plot_plsda_samples` *Plot PLSDA predictions*

Description

Plot PLSDA predictions

Usage

```
plot_plsda_samples(model, newdata = NULL, plot = TRUE)
```

Arguments

`model`	A plsda model
`newdata`	newdata to predict, if not included model$X_test will be used
`plot`	A boolean that indicate if results are plotted or not

Value

A plot of the samples or a ggplot object

Examples

```
#' # Data analysis for a table of integrated peaks

## Generate an artificial nmr_dataset_peak_table:
### Generate artificial metadata:
num_samples <- 32 # use an even number in this example
num_peaks <- 20
metadata <- data.frame(
    NMRExperiment = as.character(1:num_samples),
    Condition = rep(c("A", "B"), times = num_samples / 2)
)

### The matrix with peaks
peak_means <- runif(n = num_peaks, min = 300, max = 600)
peak_sd <- runif(n = num_peaks, min = 30, max = 60)
peak_matrix <- mapply(function(mu, sd) rnorm(num_samples, mu, sd),
    mu = peak_means, sd = peak_sd
)
colnames(peak_matrix) <- paste0("Peak", 1:num_peaks)
```

```
## Artificial differences depending on the condition:
peak_matrix[metadata$Condition == "A", "Peak2"] <-
    peak_matrix[metadata$Condition == "A", "Peak2"] + 70

peak_matrix[metadata$Condition == "A", "Peak6"] <-
    peak_matrix[metadata$Condition == "A", "Peak6"] - 60

### The nmr_dataset_peak_table
peak_table <- new_nmr_dataset_peak_table(
    peak_table = peak_matrix,
    metadata = list(external = metadata)
)

## We will use a double cross validation, splitting the samples with random
## subsampling both in the external and internal validation.
## The classification model will be a PLSDA, exploring at maximum 3 latent
## variables.
## The best model will be selected based on the area under the ROC curve
methodology <- plsda_auroc_vip_method(ncomp = 1)
model <- nmr_data_analysis(
    peak_table,
    y_column = "Condition",
    identity_column = NULL,
    external_val = list(iterations = 1, test_size = 0.25),
    internal_val = list(iterations = 1, test_size = 0.25),
    data_analysis_method = methodology
)

# plot_plsda_samples(model$outer_cv_results[[1]]$model)
```

`plot_vip_scores` *Plot vip scores of bootstrap*

Description

Plot vip scores of bootstrap

Usage

```
plot_vip_scores(vip_means, error, nbootstrap, plot = TRUE)
```

Arguments

`vip_means`	vips means values of bootstraps
`error`	error tolerated, calculated in the bootstrap
`nbootstrap`	number of bootstraps realiced
`plot`	A boolean that indicate if results are plotted or not

Value

A plot of the results or a ggplot object

Examples

```
# Data analysis for a table of integrated peaks

## Generate an artificial nmr_dataset_peak_table:
### Generate artificial metadata:
num_samples <- 64 # use an even number in this example
num_peaks <- 20
metadata <- data.frame(
    NMRExperiment = as.character(1:num_samples),
    Condition = rep(c("A", "B"), times = num_samples / 2)
)

### The matrix with peaks
peak_means <- runif(n = num_peaks, min = 300, max = 600)
peak_sd <- runif(n = num_peaks, min = 30, max = 60)
peak_matrix <- mapply(function(mu, sd) rnorm(num_samples, mu, sd),
    mu = peak_means, sd = peak_sd
)
colnames(peak_matrix) <- paste0("Peak", 1:num_peaks)

## Artificial differences depending on the condition:
peak_matrix[metadata$Condition == "A", "Peak2"] <-
    peak_matrix[metadata$Condition == "A", "Peak2"] + 70

peak_matrix[metadata$Condition == "A", "Peak6"] <-
    peak_matrix[metadata$Condition == "A", "Peak6"] - 60

### The nmr_dataset_peak_table
peak_table <- new_nmr_dataset_peak_table(
    peak_table = peak_matrix,
    metadata = list(external = metadata)
)

## We will use bootstrap and permutation method for VIPs selection
## in a a k-fold cross validation
# bp_results <- bp_kfold_VIP_analysis(peak_table, # Data to be analized
#                             y_column = "Condition", # Label
#                             k = 3,
#                             ncomp = 1,
#                             nbootstrap = 10)

# message("Selected VIPs are: ", bp_results$importarn_vips)

# plot_vip_scores(bp_results$kfold_results[[1]]$vip_means,
#                 bp_results$kfold_results[[1]]$error[1],
#                 nbootstrap = 10)
```

plot_webgl *Plot a dataset into a HTML file*

Description

Uses WebGL for performance

Usage

```
plot_webgl(nmr_dataset, html_filename, overwrite = NULL, ...)
```

Arguments

nmr_dataset	An nmr_dataset_1D
html_filename	The output HTML filename to be created
overwrite	Overwrite the lib/ directory (use NULL to prompt the user)
...	Arguments passed on to plot.nmr_dataset_1D
	x a nmr_dataset_1D object
	chemshift_range range of the chemical shifts to be included. Can be of length 3 to include the resolution in the third element (e.g. c(0.2, 0.8, 0.005))
	NMRExperiment A character vector with the NMRExperiments to include. Use "all" to include all experiments.
	quantile_plot If TRUE plot the 10\ If two numbers between 0 and 1 are given then a custom percentile can be plotted
	quantile_colors A vector with the colors for each of the quantiles
	interactive if TRUE return an interactive plotly plot, otherwise return a ggplot one.

Value

the html filename created

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
# dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
# dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
# html_plot <- plot_webgl(dataset_1D, "html_plot.html")
```

plsda_auroc_vip_compare *Compare PLSDA auroc VIP results*

Description

Compare PLSDA auroc VIP results

Usage

```
plsda_auroc_vip_compare(...)
```

Arguments

...	Results of nmr_data_analysis to be combined. Give each result a name.

Value

A plot of the AUC for each method

Examples

```
# Data analysis for a table of integrated peaks

## Generate an artificial nmr_dataset_peak_table:
### Generate artificial metadata:
num_samples <- 32 # use an even number in this example
num_peaks <- 20
metadata <- data.frame(
    NMRExperiment = as.character(1:num_samples),
    Condition = rep(c("A", "B"), times = num_samples / 2)
)

### The matrix with peaks
peak_means <- runif(n = num_peaks, min = 300, max = 600)
peak_sd <- runif(n = num_peaks, min = 30, max = 60)
peak_matrix <- mapply(function(mu, sd) rnorm(num_samples, mu, sd),
    mu = peak_means, sd = peak_sd
)
colnames(peak_matrix) <- paste0("Peak", 1:num_peaks)

## Artificial differences depending on the condition:
peak_matrix[metadata$Condition == "A", "Peak2"] <-
    peak_matrix[metadata$Condition == "A", "Peak2"] + 70

peak_matrix[metadata$Condition == "A", "Peak6"] <-
    peak_matrix[metadata$Condition == "A", "Peak6"] - 60

### The nmr_dataset_peak_table
peak_table <- new_nmr_dataset_peak_table(
```

```
    peak_table = peak_matrix,
    metadata = list(external = metadata)
)

## We will use a double cross validation, splitting the samples with random
## subsampling both in the external and internal validation.
## The classification model will be a PLSDA, exploring at maximum 3 latent
## variables.
## The best model will be selected based on the area under the ROC curve
methodology <- plsda_auroc_vip_method(ncomp = 1)
model1 <- nmr_data_analysis(
    peak_table,
    y_column = "Condition",
    identity_column = NULL,
    external_val = list(iterations = 1, test_size = 0.25),
    internal_val = list(iterations = 1, test_size = 0.25),
    data_analysis_method = methodology
)

methodology2 <- plsda_auroc_vip_method(ncomp = 2)
model2 <- nmr_data_analysis(
    peak_table,
    y_column = "Condition",
    identity_column = NULL,
    external_val = list(iterations = 1, test_size = 0.25),
    internal_val = list(iterations = 1, test_size = 0.25),
    data_analysis_method = methodology2
)

plsda_auroc_vip_compare(model1 = model1, model2 = model2)
```

`plsda_auroc_vip_method` *Method for nmr_data_analysis (PLSDA model with AUROC and VIP outputs)*

Description

Method for nmr_data_analysis (PLSDA model with AUROC and VIP outputs)

Usage

```
plsda_auroc_vip_method(ncomp, auc_increment_threshold = 0.05)
```

Arguments

`ncomp`	Max. number of latent variables to explore in the PLSDA analysis
`auc_increment_threshold`	Choose the number of latent variables when the AUC does not increment more than this threshold.

Value

Returns an object to be used with nmr_data_analysis to perform a (optionally multilevel) PLS-DA model, using the area under the ROC curve as figure of merit to determine the optimum number of latent variables.

Examples

```
method <- plsda_auroc_vip_method(3)
```

`ppm_resolution` *Unlisted PPM resolution*

Description

A wrapper to unlist the output from the function `nmr_ppm_resolution(nmr_dataset)` when no interpolation has been applied.

Usage

```
ppm_resolution(nmr_dataset)
```

Arguments

`nmr_dataset`	An object containing NMR samples

Value

A number (the ppm resolution, measured in ppms)

Numeric (the ppm resolution, measured in ppms)

Examples

```
nmr_dataset <- nmr_dataset_load(system.file("extdata", "nmr_dataset.rds", package = "AlpsNMR"))
nmr_ppm_resolution(nmr_dataset)
```

print.nmr_dataset *Print for nmr_dataset*

Description

Print for nmr_dataset

Usage

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

Arguments

x	an nmr_dataset object
...	for future use

Value

Print for nmr_dataset

See Also

Other class helper functions: format.nmr_dataset(), format.nmr_dataset_1D(), format.nmr_dataset_peak_table(), is.nmr_dataset_1D(), is.nmr_dataset_peak_table(), new_nmr_dataset(), new_nmr_dataset_1D(), new_nmr_dataset_peak_table(), print.nmr_dataset_1D(), print.nmr_dataset_peak_table(), validate_nmr_dataset(), validate_nmr_dataset_family(), validate_nmr_dataset_peak_table()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
print(dataset)
```

print.nmr_dataset_1D *print for nmr_dataset_1D*

Description

print for nmr_dataset_1D

Usage

```
## S3 method for class 'nmr_dataset_1D'
print(x, ...)
```

Arguments

x	an nmr_dataset_1D object
...	for future use

Value

print for nmr_dataset_1D

See Also

Other class helper functions: format.nmr_dataset(), format.nmr_dataset_1D(), format.nmr_dataset_peak_table(), is.nmr_dataset_1D(), is.nmr_dataset_peak_table(), new_nmr_dataset(), new_nmr_dataset_1D(), new_nmr_dataset_peak_table(), print.nmr_dataset(), print.nmr_dataset_peak_table(), validate_nmr_dataset(), validate_nmr_dataset_family(), validate_nmr_dataset_peak_table()

Other nmr_dataset_1D functions: [.nmr_dataset_1D(), format.nmr_dataset_1D(), get_integration_with_metadata, is.nmr_dataset_1D(), nmr_integrate_peak_positions(), nmr_integrate_regions(), nmr_meta_add(), nmr_meta_export(), nmr_meta_get(), nmr_meta_get_column(), nmr_ppm_resolution()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
print(dataset_1D)
```

---

print.nmr_dataset_peak_table

print for nmr_dataset_peak_table

Description

print for nmr_dataset_peak_table

Usage

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

Arguments

x	an nmr_dataset_peak_table object
...	for future use

Value

print for nmr_dataset_peak_table

See Also

Other class helper functions: format.nmr_dataset(), format.nmr_dataset_1D(), format.nmr_dataset_peak_table(), is.nmr_dataset_1D(), is.nmr_dataset_peak_table(), new_nmr_dataset(), new_nmr_dataset_1D(), new_nmr_dataset_peak_table(), print.nmr_dataset(), print.nmr_dataset_1D(), validate_nmr_dataset(), validate_nmr_dataset_family(), validate_nmr_dataset_peak_table()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
meta <- file.path(dir_to_demo_dataset, "dummy_metadata.xlsx")
metadata <- readxl::read_excel(meta, sheet = 1)
dataset_1D <- nmr_meta_add(dataset_1D, metadata = metadata, by = "NMRExperiment")
metadata <- list(external = dataset_1D[["metadata"]][["external"]])
peak_table <- nmr_data(dataset_1D)
new <- new_nmr_dataset_peak_table(peak_table, metadata)
new
```

---

random_subsampling *Random subsampling*

Description

Random subsampling

Usage

```
random_subsampling(
  sample_idx,
  iterations = 10L,
  test_size = 0.25,
  keep_together = NULL,
  balance_in_train = NULL
)
```

Arguments

sample_idx	Typically a numeric vector with sample index to be separated. A character vector with sample IDs could also be used
iterations	An integer, the number of iterations in the random subsampling
test_size	A number between 0 and 1. The samples to be included in the test set on each interation.
keep_together	Either NULL or a factor with the same length as sample_idx. keep_together can be used to ensure that groups of samples are kept in together in all iterations (either on training or on test, but never split). A typical use case for this is when you have sample replicates and you want to keep all replicates together

to prevent overoptimistic results (having one sample on the train subset and its replicate on the test subset would make the prediction easier to guess). Another use case for this is when you have a longitudinal study and you want to keep some subjects in the same train or test group, because you want to use some information in a longitudinal way (e.g. a multilevel plsda model).

`balance_in_train`

Either `NULL` or a factor with the same length as `sample_idx`. `balance_in_train` can be used to force that on each iteration, the train partition contains the same number of samples of the given factor levels. For instance, if we have a dataset with 40 samples of class "A" and 20 samples of class "B", using a `test_size = 0.25`, we can force to always have 16 samples of class "A" and 16 samples of class "B" in the training subset. This is beneficial to those algorithms that require that the training groups are balanced.

Value

A list of length equal to `iterations`. Each element of the list is a list with two entries (`training` and `test`) containing the `sample_idx` values that will belong to each subset.

Examples

```
random_subsampling(1:100, iterations = 4, test_size = 0.25)

subject_id <- c("Alice", "Bob", "Charlie", "Eve")
random_subsampling(1:4, iterations = 2, test_size = 0.25, keep_together = subject_id)
```

ROI_blood *ROIs for blood (plasma/serum) samples*

Description

The template ROI_blood contains the targeted list of metabolites to be quantified (blood samples)

References

github.com/danielcanueto/rDolphin

Examples

```
data("ROI_blood")
ROI_blood[ROI_blood$Metabolite == "Valine", ]
```

ROI_cell *ROIs for cell samples*

Description

The template ROI_cell contains the targeted list of metabolites to be quantified (cell samples)

References

github.com/danielcanueto/rDolphin

Examples

```
data("ROI_cell")
ROI_cell[ROI_cell$Metabolite == "Valine", ]
```

ROI_urine *ROIs for urine samples*

Description

The template ROI_urine contains the targeted list of metabolites to be quantified (urine samples)

References

github.com/danielcanueto/rDolphin

Examples

```
data("ROI_urine")
ROI_urine[ROI_urine$Metabolite == "Valine", ]
```

`save_files_to_rDolphin` *Save files to rDoplhin*

Description

The function saves the CSV files required by to_rDolphin and Automatic_targeted_profiling functions for metabolite profiling.

Usage

```
save_files_to_rDolphin(files_rDolphin, output_directory)
```

Arguments

`files_rDolphin` a list containing 4 elements from `files_to_rDolphin`

- `meta_rDolphin`: metadata in rDolphin format,
- `NMR_spectra`: spectra matrix
- `ROI`: ROI template
- `Parameters_blood`: parameters file

`output_directory` a directory in which the CSV files are saved

Value

CSV files containing:

See Also

Other import/export functions: `Pipelines`, `files_to_rDolphin()`, `load_and_save_functions`, `nmr_data()`, `nmr_meta_export()`, `nmr_read_bruker_fid()`, `nmr_read_samples()`, `nmr_zip_bruker_samples()`, `save_profiling_output()`, `to_ChemoSpec()`

Examples

```
## Not run:
dataset <- system.file("dataset-demo", package = "AlpsNMR")
excel_file <- system.file("dataset-demo", "dummy_metadata.xlsx", package = "AlpsNMR")
nmr_dataset <- nmr_read_samples_dir(dataset)
files_rDolphin <- files_to_rDolphin_blood(nmr_dataset)
save_files_to_rDolphin(files_rDolphin, output_directory = ".")

## End(Not run)
```

`save_profiling_output` *Save rDoplhin output*

Description

The function saves the output from Automatic_targeted_profiling function in CSV format.

Usage

```
save_profiling_output(targeted_profiling, output_directory)
```

Arguments

`targeted_profiling`	A list from Automatic_targeted_profiling function
`output_directory`	a directory in which the CSV files are saved

Value

rDolphin output from Automatic_targeted_profiling function:

- metabolites_intensity
- metabolites_quantification
- ROI_profiles_used
- chemical_shift
- fitting_error
- half_bandwidth
- signal_area_ratio

See Also

Other import/export functions: `Pipelines`, `files_to_rDolphin()`, `load_and_save_functions`, `nmr_data()`, `nmr_meta_export()`, `nmr_read_bruker_fid()`, `nmr_read_samples()`, `nmr_zip_bruker_samples()`, `save_files_to_rDolphin()`, `to_ChemoSpec()`

Examples

```
## Not run:
rDolphin_object <- to_rDolphin(parameters)
targeted_profiling <- Automatic_targeted_profiling(rDolphin)
save_profiling_output(targeted_profiling, output_directory)

## End(Not run)
```

SummarizedExperiment_to_nmr_dataset_peak_table

Import SummarizedExperiment as mr_dataset_peak_table

Description

Import SummarizedExperiment as mr_dataset_peak_table

Usage

```
SummarizedExperiment_to_nmr_dataset_peak_table(se)
```

Arguments

se	An SummarizedExperiment object

Value

nmr_dataset_peak_table An nmr_dataset_peak_table object (unmodified)

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
meta <- file.path(dir_to_demo_dataset, "dummy_metadata.xlsx")
metadata <- readxl::read_excel(meta, sheet = 1)
dataset_1D <- nmr_meta_add(dataset_1D, metadata = metadata, by = "NMRExperiment")
metadata <- list(external = dataset_1D[["metadata"]][["external"]])
peak_table <- nmr_data(dataset_1D)
nmr_peak_table <- new_nmr_dataset_peak_table(peak_table, metadata)
se <- nmr_dataset_peak_table_to_SummarizedExperiment(nmr_peak_table)
nmr_peak_table <- SummarizedExperiment_to_nmr_dataset_peak_table(se)
```

SummarizedExperiment_to_nmr_data_1r

Import SummarizedExperiment as 1D NMR data

Description

Import SummarizedExperiment as 1D NMR data

Usage

```
SummarizedExperiment_to_nmr_data_1r(se)
```

Arguments

se	An SummarizedExperiment object

Value

nmr_dataset An nmr_dataset_1D object (unmodified)

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
se <- nmr_data_1r_to_SummarizedExperiment(dataset_1D)
dataset_1D <- SummarizedExperiment_to_nmr_data_1r(se)
```

---

tidy.nmr_dataset_1D *Get a tidy data frame from nmr_data object*

Description

This dataframe is useful for plotting with ggplot, although it may be very long and therefore use a lot of RAM.

Usage

```
## S3 method for class 'nmr_dataset_1D'
tidy(
  x,
  NMRExperiment = NULL,
  chemshift_range = NULL,
  columns = character(0L),
  matrix_name = "data_1r",
  axis_name = "axis",
  ...
)
```

Arguments

x	an nmr_dataset_1D object
NMRExperiment	A character vector with the NMRExperiments to include. NULL means all.
chemshift_range	range of the chemical shifts to be included. Can be of length 3 to include the resolution in the third element (e.g. c(0.2, 0.8, 0.005))
columns	A character vector with the metadata columns to get, use NULL to get all of them.
matrix_name	A string with the matrix name, typically "data_1r"
axis_name	A string with the axis name, for now "axis" is the only valid option
...	Ignored

Value

A data frame with `NMRExperiment`, `chemshift`, `intensity` and any additional column requested

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -1.0, max = 1.6, by = 2.3E-4))
dummy_metadata <- system.file("dataset-demo", "dummy_metadata.xlsx", package = "AlpsNMR")
NMRExp_SubjID <- readxl::read_excel(dummy_metadata, sheet = 1)
dataset_1D <- nmr_meta_add(dataset_1D, NMRExp_SubjID)
df_for_ggplot <- tidy(dataset_1D, chemshift_range = c(1.2, 1.4), columns = "SubjectID")
head(df_for_ggplot)
```

---

to_ASICS *Export data for the ASICS spectral quantification library*

Description

Exports the spectra matrix, sample names and chemical shift axis into an ASICS Spectra object.

Usage

```
to_ASICS(dataset, ...)
```

Arguments

`dataset`	An nmr_dataset_1D object
`...`	Arguments passed on to `ASICS::createSpectra`
	`norm.method` Character specifying the normalisation method to use on spectra ONLY if the `importSpectra` function was not used.
	`norm.params` List containing normalisation parameteres (see `normaliseSpectra` for details) ONLY if the `importSpectra` function was not used.

Value

An ASICS::Spectra object

Examples

```
if (requireNamespace("ASICS", quietly=TRUE)) {
  nsamp <- 3
  npoints <- 300
  metadata <- list(external = data.frame(
    NMRExperiment = paste0("Sample", seq_len(nsamp))
  ))
  dataset <- new_nmr_dataset_1D(
    ppm_axis = seq(from = 0.2, to = 10, length.out = npoints),
```

```
    data_1r = matrix(runif(nsamp * npoints), nrow = nsamp, ncol = npoints),
    metadata = metadata
  )
  forAsics <- to_ASICS(dataset)
  #ASICS::ASICS(forAsics)
}
```

`to_ChemoSpec` *Convert to ChemoSpec Spectra class*

Description

Convert to ChemoSpec Spectra class

Usage

```
to_ChemoSpec(nmr_dataset, desc = "A nmr_dataset", group = NULL)
```

Arguments

`nmr_dataset`	An nmr_dataset_1D object
`desc`	a description for the dataset
`group`	A string with the column name from the metadata that has grouping information

Value

A Spectra object from the ChemoSpec package

See Also

Other import/export functions: `Pipelines`, `files_to_rDolphin()`, `load_and_save_functions`, `nmr_data()`, `nmr_meta_export()`, `nmr_read_bruker_fid()`, `nmr_read_samples()`, `nmr_zip_bruker_samples()`, `save_files_to_rDolphin()`, `save_profiling_output()`

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
chemo_spectra <- to_ChemoSpec(dataset_1D)
```

`validate_nmr_dataset` *Validate nmr_dataset objects*

Description

Validate nmr_dataset objects

Validate 1D nmr datasets

Usage

```
validate_nmr_dataset(samples)

validate_nmr_dataset_1D(nmr_dataset_1D)
```

Arguments

`samples`	An nmr_dataset object
`nmr_dataset_1D`	An nmr_dataset_1D object

Value

Validate nmr_dataset objects

The nmr_dataset_1D unchanged

This function is useful for its side-effects. Stopping in case of error

See Also

Other class helper functions: `format.nmr_dataset()`, `format.nmr_dataset_1D()`, `format.nmr_dataset_peak_table()`, `is.nmr_dataset_1D()`, `is.nmr_dataset_peak_table()`, `new_nmr_dataset()`, `new_nmr_dataset_1D()`, `new_nmr_dataset_peak_table()`, `print.nmr_dataset()`, `print.nmr_dataset_1D()`, `print.nmr_dataset_peak_table()`, `validate_nmr_dataset_family()`, `validate_nmr_dataset_peak_table()`

Other class helper functions: `format.nmr_dataset()`, `format.nmr_dataset_1D()`, `format.nmr_dataset_peak_table()`, `is.nmr_dataset_1D()`, `is.nmr_dataset_peak_table()`, `new_nmr_dataset()`, `new_nmr_dataset_1D()`, `new_nmr_dataset_peak_table()`, `print.nmr_dataset()`, `print.nmr_dataset_1D()`, `print.nmr_dataset_peak_table()`, `validate_nmr_dataset_family()`, `validate_nmr_dataset_peak_table()`

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
validate_nmr_dataset(dataset)

dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
dataset_1D_validated <- validate_nmr_dataset_1D(dataset_1D)
```

validate_nmr_dataset_family *Validate nmr_dataset_family objects*

Description

Validate nmr_dataset_family objects

Usage

```
validate_nmr_dataset_family(nmr_dataset_family)
```

Arguments

nmr_dataset_family	An nmr_dataset_family object

Value

The nmr_dataset_family unchanged

This function is useful for its side-effects: Stopping in case of error

See Also

Other class helper functions: format.nmr_dataset(), format.nmr_dataset_1D(), format.nmr_dataset_peak_table(), is.nmr_dataset_1D(), is.nmr_dataset_peak_table(), new_nmr_dataset(), new_nmr_dataset_1D(), new_nmr_dataset_peak_table(), print.nmr_dataset(), print.nmr_dataset_1D(), print.nmr_dataset_peak_table(), validate_nmr_dataset(), validate_nmr_dataset_peak_table()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
validate_nmr_dataset_family(dataset_1D)
```

validate_nmr_dataset_peak_table *Validate nmr_dataset_peak_table objects*

Description

Validate nmr_dataset_peak_table objects

Usage

```
validate_nmr_dataset_peak_table(nmr_dataset_peak_table)
```

Arguments

nmr_dataset_peak_table
: An nmr_dataset_peak_table object

Value

The nmr_dataset_peak_table unchanged

See Also

Other class helper functions: format.nmr_dataset(), format.nmr_dataset_1D(), format.nmr_dataset_peak_table(), is.nmr_dataset_1D(), is.nmr_dataset_peak_table(), new_nmr_dataset(), new_nmr_dataset_1D(), new_nmr_dataset_peak_table(), print.nmr_dataset(), print.nmr_dataset_1D(), print.nmr_dataset_peak_table(), validate_nmr_dataset(), validate_nmr_dataset_family()

Examples

```
pt <- new_nmr_dataset_peak_table(
   peak_table = matrix(c(1, 2), nrow = 1, dimnames = list("10", c("ppm_1.4", "ppm_1.6"))),
    metadata = list(external = data.frame(NMRExperiment = "10"))
)
pt_validated <- validate_nmr_dataset_peak_table(pt)
```

[.nmr_dataset *Extract parts of an nmr_dataset*

Description

Extract parts of an nmr_dataset

Usage

```
## S3 method for class 'nmr_dataset'
x[i]
```

Arguments

x
: an nmr_dataset object

i
: indices of the samples to keep

Value

an nmr_dataset with the extracted samples

See Also

Other subsetting functions: [.nmr_dataset_1D(), [.nmr_dataset_peak_table(), filter.nmr_dataset_family(), nmr_pca_outliers_filter()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset2 <- dataset[1:3] # get the first 3 samples
```

[.nmr_dataset_1D *Extract parts of an nmr_dataset_1D*

Description

Extract parts of an nmr_dataset_1D

Usage

```
## S3 method for class 'nmr_dataset_1D'
x[i]
```

Arguments

x	an nmr_dataset_1D object
i	indices of the samples to keep

Value

an nmr_dataset_1D with the extracted samples

See Also

Other subsetting functions: [.nmr_dataset(), [.nmr_dataset_peak_table(), filter.nmr_dataset_family(), nmr_pca_outliers_filter()

Other nmr_dataset_1D functions: format.nmr_dataset_1D(), get_integration_with_metadata(), is.nmr_dataset_1D(), nmr_integrate_peak_positions(), nmr_integrate_regions(), nmr_meta_add(), nmr_meta_export(), nmr_meta_get(), nmr_meta_get_column(), nmr_ppm_resolution(), print.nmr_dataset_1D()

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
dataset_1D[0]
```

`[.nmr_dataset_peak_table` *Extract parts of an nmr_dataset_peak_table*

Description

Extract parts of an nmr_dataset_peak_table

Usage

```
## S3 method for class 'nmr_dataset_peak_table'
x[i]
```

Arguments

x	an nmr_dataset_peak_table object
i	indices of the samples to keep

Value

an nmr_dataset_peak_table with the extracted samples

See Also

Other subsetting functions: `[.nmr_dataset()`, `[.nmr_dataset_1D()`, `filter.nmr_dataset_family()`, `nmr_pca_outliers_filter()`

Examples

```
dir_to_demo_dataset <- system.file("dataset-demo", package = "AlpsNMR")
dataset <- nmr_read_samples_dir(dir_to_demo_dataset)
dataset_1D <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10, by = 2.3E-4))
meta <- file.path(dir_to_demo_dataset, "dummy_metadata.xlsx")
metadata <- readxl::read_excel(meta, sheet = 1)
dataset_1D <- nmr_meta_add(dataset_1D, metadata = metadata, by = "NMRExperiment")
metadata <- list(external = dataset_1D[["metadata"]][["external"]])
peak_table <- nmr_data(dataset_1D)
new <- new_nmr_dataset_peak_table(peak_table, metadata)
new[0]
```

Index