NMR-based metabolomic analysis of the dataset MTBLS242: serum samples

IBEC

October 12, 2022,15:33

This is an example of using AlpsNMR package on the MTBLS242 dataset structured as a pipeline so that inputs are needed and outputs are obtained in the selected folders. Edit "inputs" to match your parameters and just run the "code to run" for the pipeline execution. However, you can see the vignettes and use the functions as you wish. You can download the MTBLS242 dataset from MetaboLights database: https://www.ebi.ac.uk/metabolights/MTBLS242

```
library(AlpsNMR)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
## Loading required package: future
```

```
## Loading required package: magrittr
```

```
library(BiocParallel)
```

Pipeline preparation

To work as in a pipeline manner we need to set an output directory. We can set the number of cores of your computer for parallelization.

```
# Set a folder to keep results
output_dir <- "C:/Users/Biosignal/Desktop/Seminario_metabolomics/results"

# How many cores to use for parallelization
num_workers <- 6
```

Node 1: Load samples

Loads samples from a specified directory into a `nmr_dataset` object. Then we can save the loaded data into the output directory.

Input parameters

```
# Path of NMR samples downloaded from https://www.ebi.ac.uk/metabolights/MTBLS242:
dataset_path_nmr <- "C:/Users/Biosignal/Desktop/Seminario_metabolomics/MTBLS242"

# Files/directories ending in "s" corresponding to the spectra in the dataset:
filename_glob <- "*s"
```

Code to run

```
register(SnowParam((num_workers), progressbar = TRUE))

NMRExperiments <- as.character(fs::dir_ls(dataset_path_nmr, glob = filename_glob))
nmr_dataset <- nmr_read_samples(NMRExperiments)
```

```
##   |                                                                      |
```

```
# Path where metadata is contained
excel_file <- paste0(dataset_path_nmr, "/nmr_dataset_metadata_tidy.xlsx")
```

Code to run

```
nmr_dataset <- nmr_meta_add_tidy_excel(nmr_dataset, excel_file)
```

Node 3: Interpolation

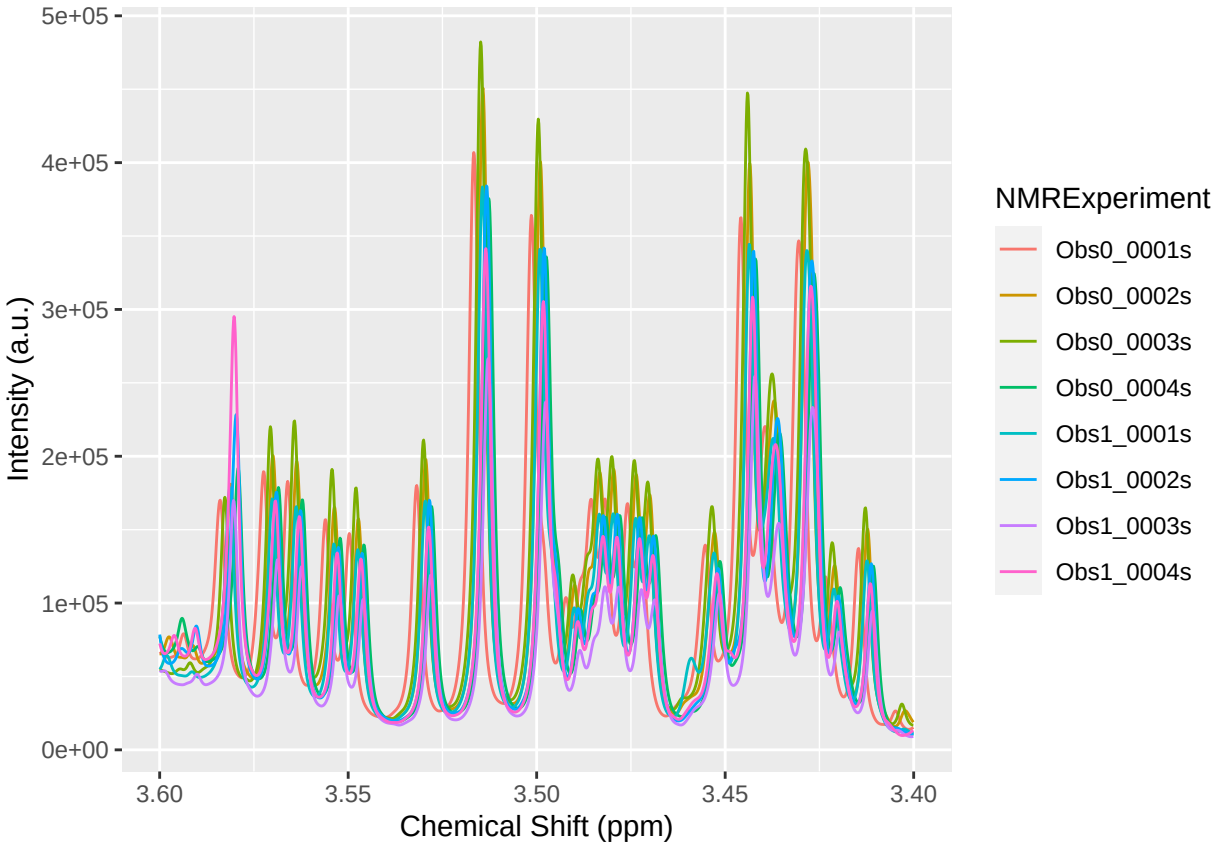
Interpolation is used to unify the ppm axis from all spectra. However, you also can set a range for next steps avoiding noise regions from here. Note that ppm resolution is automatically calculated with the function `nmr_ppm_resolution` in "Code to run".

Input parameters

```
ppm_range_start <- 0.7
ppm_range_end <- 10
```

Code to run

```
ppm_resolution <- unlist(nmr_ppm_resolution(nmr_dataset[1]))
axis <- c(min = ppm_range_start, max = ppm_range_end, by = ppm_resolution)
nmr_dataset <- nmr_interpolate_1D(nmr_dataset, axis = axis)

plot(nmr_dataset,
  NMRExperiment = c(
  "Obs0_0001s",
  "Obs0_0002s",
  "Obs0_0003s",
  "Obs0_0004s",
  "Obs1_0001s",
  "Obs1_0002s",
  "Obs1_0003s",
  "Obs1_0004s"),
  chemshift_range = c(3.40, 3.60))
```



Node 4: Region Exclusion

Here it is important to know what type of signals can mask the results due to their intensity or what type of solvent has been used in sample processing since this can create artifacts in the spectra and should be removed. In this case, the biological samples correspond to serum, which contains a lot of water and its signal should be removed from further steps. To do this, we define a vector containing the range (min ppm value, max ppm value) of the water signal, but other signals can be eliminated, for example: exclude_regions <- list(water = c(4.5, 5.1), methanol = c(3.33, 3.34))

Input parameters

```
exclude_regions <-  list(water = c(4.5, 5.1))
```

Code to run

```
nmr_dataset <- nmr_exclude_region(nmr_dataset, exclude = exclude_regions)
```

Node 5: Initial Outlier Rejection

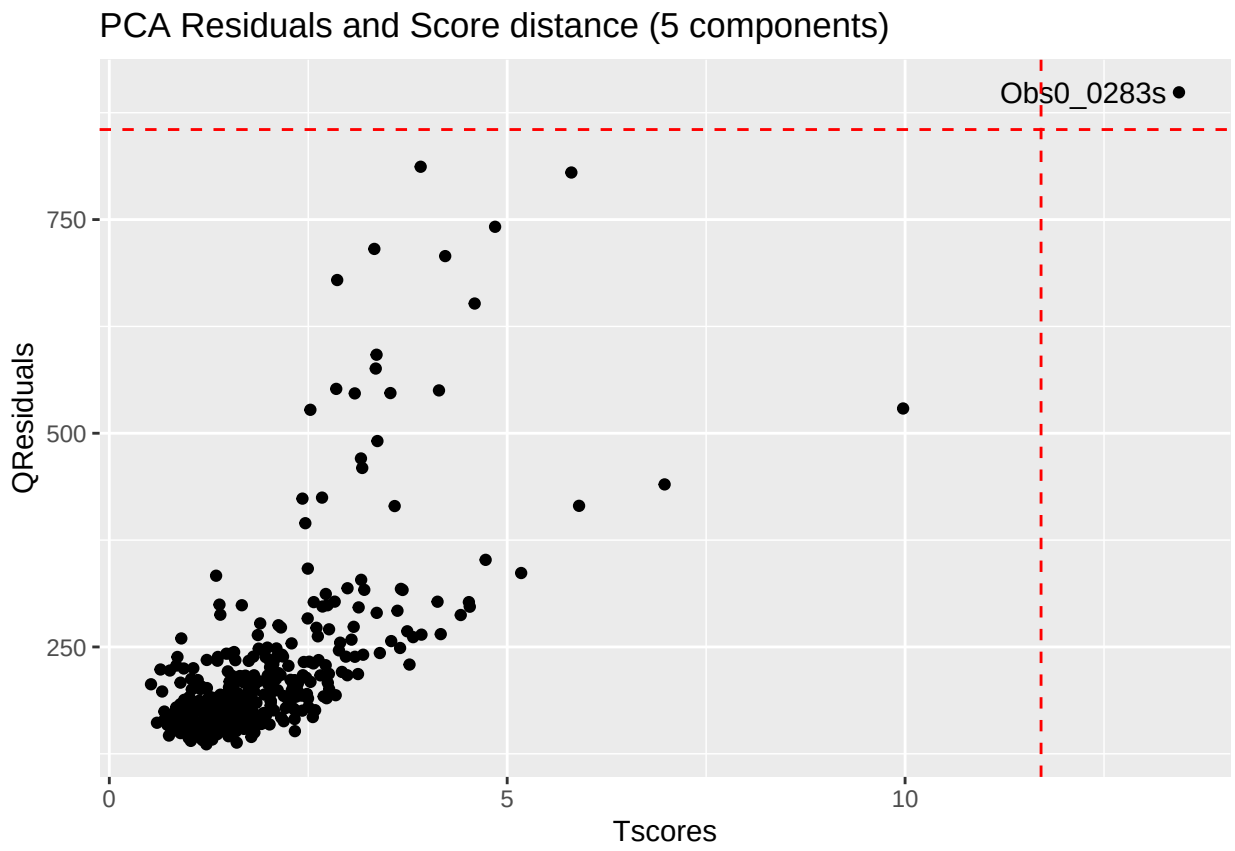
The robust principal component analysis (rPCA) for outlier detection gives an idea of potential outliers. A proposed threshold, based on quantiles, for Q residual and T2 score values, results less sensitive to extreme intensities. Then you choose if any sample should be excluded. The plot below indicated that a sample "Obs0_0283s" is extremely different than the other samples. The function is prepared to annotated samples that are in the top-right corner, exhibiting high differences.

Input parameters

```
# Nothing
```

Code to run

```
pca_outliers <- nmr_pca_outliers_robust(nmr_dataset)
nmr_pca_outliers_plot(nmr_dataset, pca_outliers)
```



Then, if we decide to discard this sample, we just run the function below. Otherwise, just ignore this:

```
nmr_dataset_with_outliers <- nmr_dataset
nmr_dataset <- nmr_pca_outliers_filter(nmr_dataset, pca_outliers)
```

Node 6: Filter samples

Input parameters

The filter node takes care of keeping only some samples. In this case, we want to compare two time points of the MTBLS242 dataset to compare them: "preop" and "3 months after surgery". However, you can filter to keep other conditions kept in the metadata. Some examples: - `Cohort == "A"`: Keeps the A cohort - `TimePoint %in% c("preop", "3 months after surgery")`: Keeps timepoints "preop" and "3 months after surgery" - `Gender == "Female"`: Keeps Female samples - others

```
samples_to_keep_conditions <- 'Timepoint %in% c("preop", "3 months after surgery")'
```

Code to run

```
conditions_expr <- rlang::parse_exprs(samples_to_keep_conditions)
nmr_dataset <- filter(nmr_dataset, !!!conditions_expr)
```

Node 7: Peak detection and Alignment

Peak detection is based on a combination of an automated baseline threshold, signal to noise ratio and maximum tolerance. Alignment is based on hierarchical cluster-based peak alignment (CluPA) (Vu et al., 2011).

Input parameters

```
# Leave those as default/recommended for serum.
# Size of peak detection segments
nDivRange_ppm <- 0.1

# Baseline threshold
baselineThresh <- NULL

# Signal to noise ratio
SNR.Th <- 10

# Maximum alignment shift
maxShift_ppm <- 0.0015
```

Code to run

```
scales <- seq(1, 16, 2)
acceptLostPeak <- FALSE

# For parallelization
plan(multisession, workers = 10)

# Step 1: Peak detection
message("Detecting peaks...")
```

```
## Detecting peaks...
```

```
peak_data <- nmr_detect_peaks(nmr_dataset,
                              nDivRange_ppm = nDivRange_ppm,
                              scales = scales,
                              baselineThresh = 0,
                              SNR.Th = SNR.Th)
```

```
## Warning: AlpsNMR now uses BiocParallel instead of future for parallellization
## i If you used plan(multisession) or other plan(), consider removing all plan() calls and use:
##     library(BiocParallel)
##     register(SnowParam(workers = 3), default = TRUE)
## i You just need to place that code once, typically at the beginning of your script
## This warning is displayed once per session.

##   |                                                                              |
```

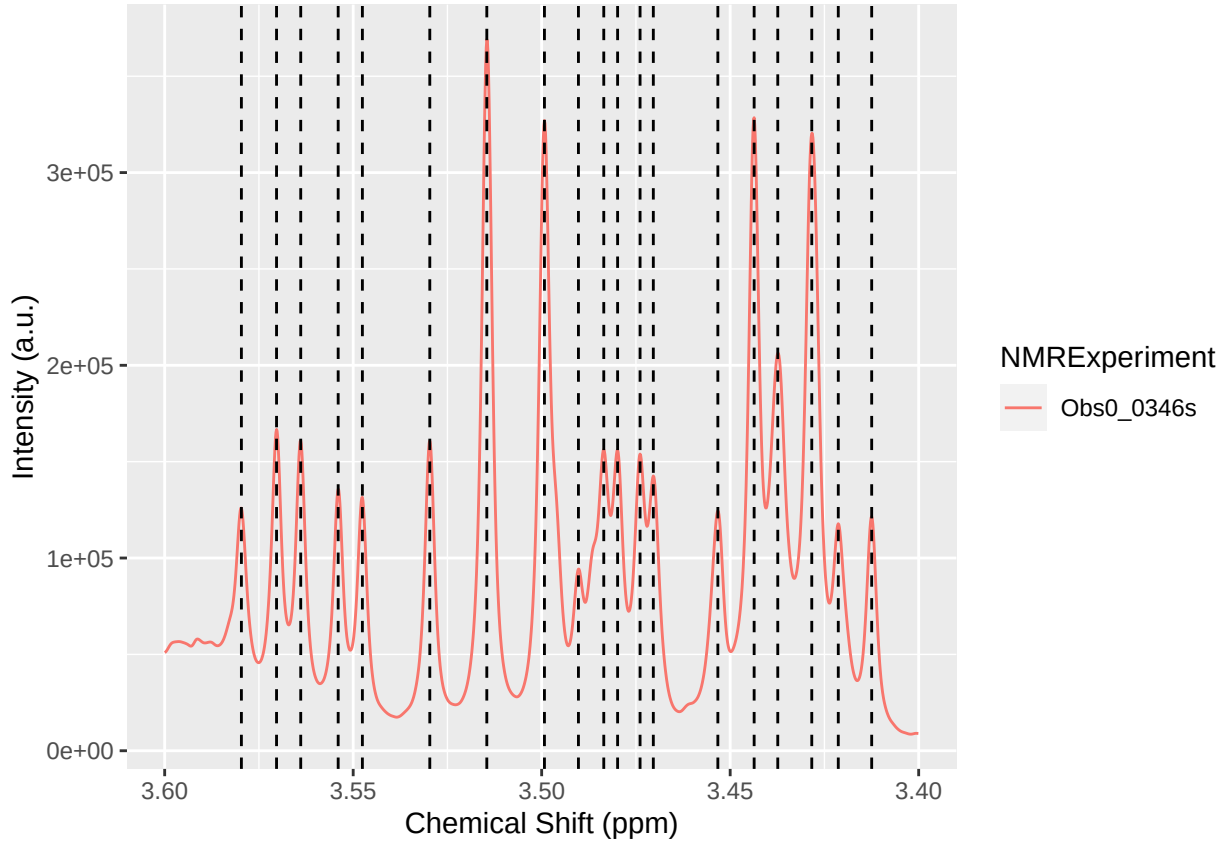
```r
# Step 2: Finding the reference spectrum for alignment
message("Choosing alignment reference...")
```

```
## Choosing alignment reference...
```

```r
NMRExp_ref <- nmr_align_find_ref(nmr_dataset, peak_data)

# Step 3: Alignment
message("Starting alignment...")
```

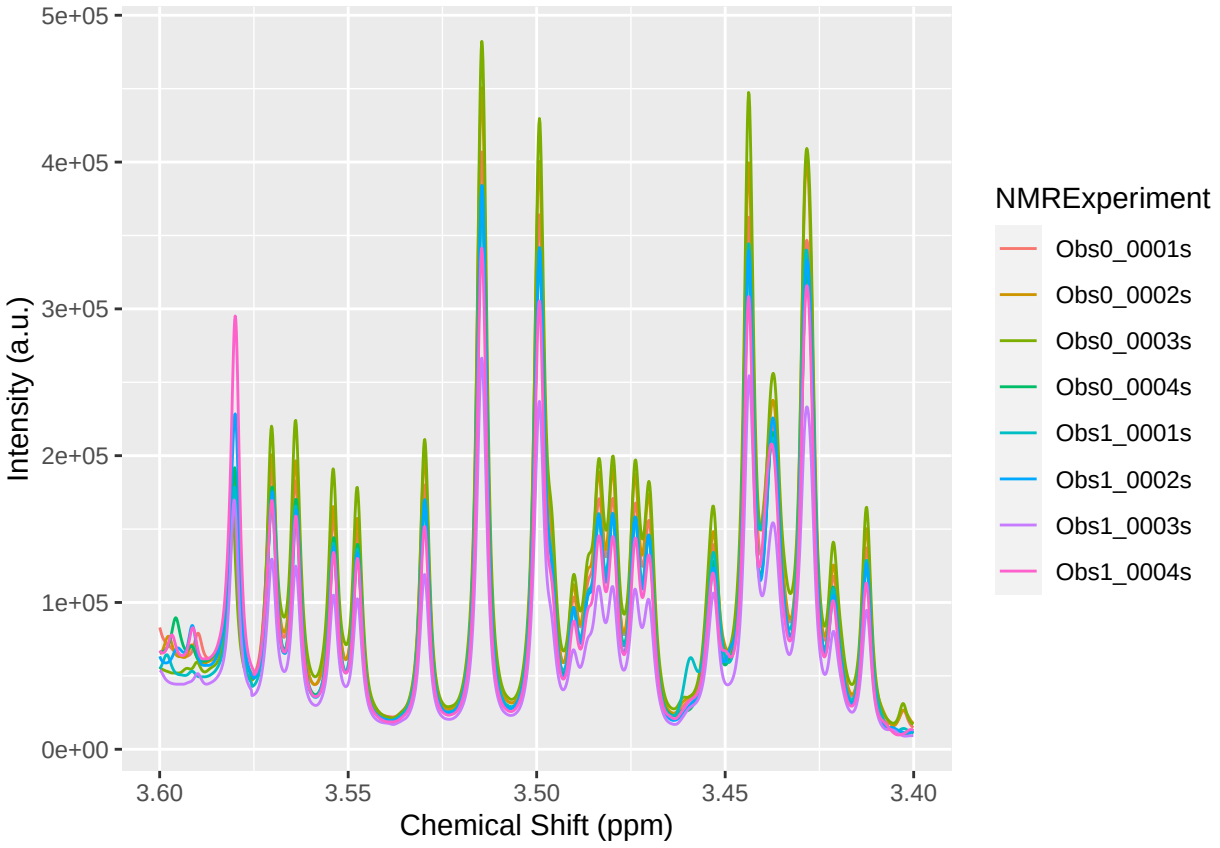
```
## Starting alignment...
```

```r
nmr_dataset <- nmr_align(nmr_dataset, peak_data,
                         NMRExp_ref = NMRExp_ref,
                         maxShift_ppm = maxShift_ppm,
                         acceptLostPeak = acceptLostPeak)

plan(sequential)

# Plotting results
nmr_detect_peaks_plot(
  nmr_dataset,
  peak_data,
  NMRExperiment = NMRExp_ref,
  chemshift_range = c(3.40, 3.60)
  )
```



we can take a look into the detected peaks. The interactive plot allows you to zoom in in HTML files.

```
plot(nmr_dataset,
  NMRExperiment = c(
  "Obs0_0001s",
  "Obs0_0002s",
  "Obs0_0003s",
  "Obs0_0004s",
  "Obs1_0001s",
  "Obs1_0002s",
  "Obs1_0003s",
  "Obs1_0004s"),
  chemshift_range = c(3.40, 3.60))
```



Node 8: Normalization

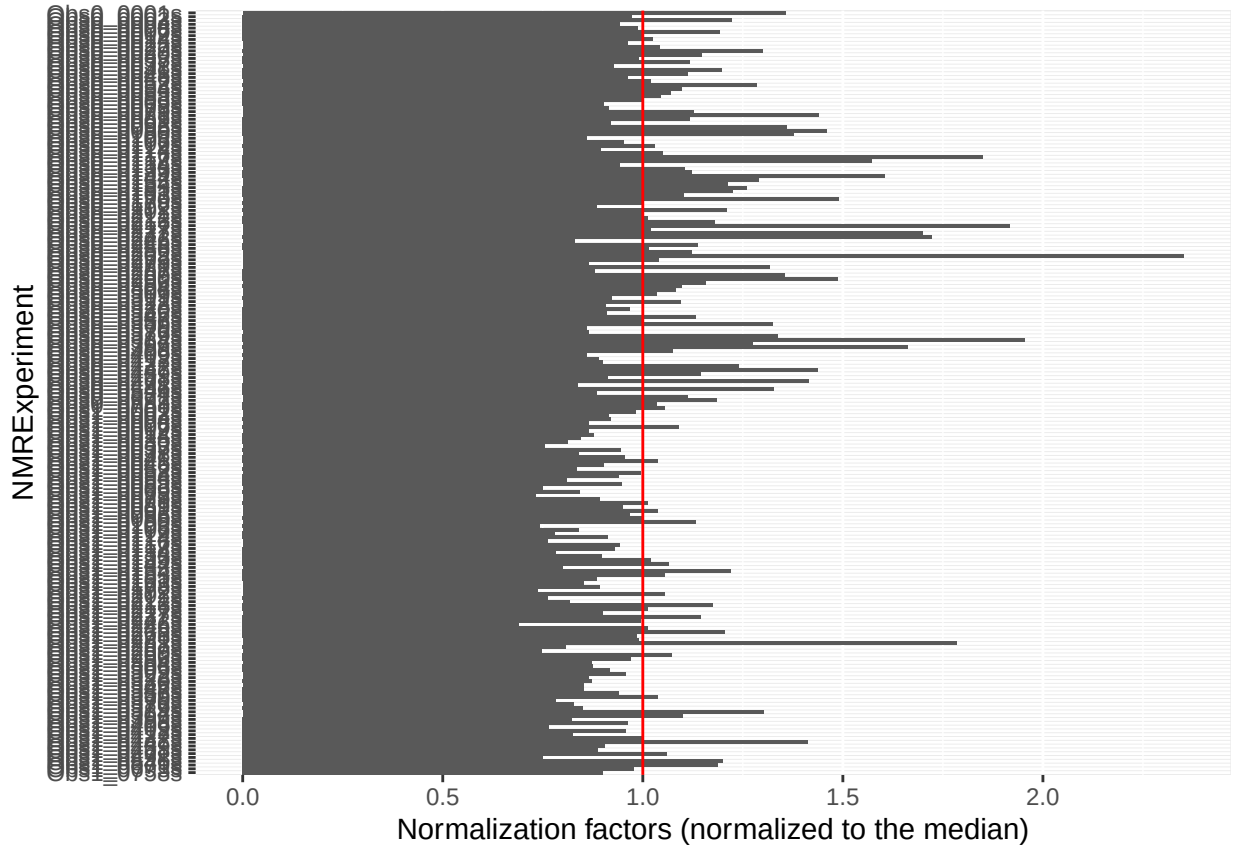
We can normalize the dataset. This is recommended for biosamples, controlling for dilution factors, irregular pipetting, etc. Probabilistic quotient normalization is one of the most used model-based techniques NMR-based metabolomics.

Input parameters

```
# nothing
```

Code to run

```
nmr_dataset <- nmr_normalize(nmr_dataset, method = "pqn")
normalization <- nmr_normalize_extra_info(nmr_dataset)
normalization$plot
```



Node 9: Integration

For peak integration, calculation of peak width may be performed automatically (set peak_width_ppm = NULL), from the detected peaks in the reference spectrum (if you wish, you can combine detected peaks other than the reference spectrum, see help), or manually, in which users can select a specific peak width for integrating the detected peaks. This differs than the bucketing approach in which spectra are equally divided into buckets (for example of 0.01 ppm) and this normally leads to a higher number of total variables. this has the inconvenient that several peaks might be split into several parts, lowering the statistical power, and vice-versa, certain overlapping tails might result in false positives because of this noisy parts. However, a good match between them is expected.

Input parameters

```
peak_width_ppm <- NULL
```

Code to run

```
# be carefull, you integrate based on peaks from a unique ref sample
peak_data_integ <- dplyr::filter(peak_data, NMRExperiment == !!NMRExp_ref)
```

```
nmr_peak_table <- nmr_integrate_peak_positions(
  samples = nmr_dataset,
  peak_pos_ppm = peak_data_integ$ppm,
  peak_width_ppm = peak_width_ppm)
```

```
## calculated width for integration is 0.00427905515639537 ppm
```

```
nmr_peak_table_completed <- get_integration_with_metadata(nmr_peak_table)
```