NMR-based metabolomic analysis by AlpsNMR of the dataset MTBLS242

Institute for Bioengineering of Catalonia

December 12, 2019

The NMR metabolomics dataset MTBLS242 was used as an example by the AlpsNMR package. The package is structured as a pipeline, so that inputs are needed, and outputs are obtained in selected folders. "Inputs" should be edited to match specific parameters and "code to run" can be initiated for pipeline execution. The vignettes can be consulted and functions can be used at wish.

```
library(AlpsNMR)
```

```
## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```

```
## Warning: replacing previous import 'ggplot2::last_plot' by
## 'plotly::last_plot' when loading 'rDolphin'
```

```
##
## Attaching package: 'AlpsNMR'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

Downloading data

The demo dataset used in this tutorial is approximately 350MB large. Due to its size it can't be bundled with the AlpsNMR package.

The MTBLS242 dataset can be downloaded directly from the public MetaboLights repository: https://www.ebi.ac.uk/metabolights/MTBLS242. Alternatively, the contents (NMR spectra and metadata) can be unzipped from a Dropbox link in the README file: https://sipss.github.io/AlpsNMR.

For convenience, we have also prepared a function (tested for WINDOWS) that downloads a zip file with all the samples and metadata from Dropbox:

```
download_demo(to = "C:/Users/fmadrid/")
```

Pipeline preparation

To work in a pipeline manner, an output directory should be created. The number of computer cores to be used for parallelization can be set.

```r
# Set a folder to save results
output_dir <- "C:/Users/fmadrid/results/"

# How many cores to use for parallelization and speed up the processing
num_workers <- 8
```

Node 1: Load samples

The samples can be loaded from a specified directory into a `nmr_dataset` object. The loaded data can be then saved into the output directory. `Plan` function can be used to activate (multiprocess) or deactivate (sequential) parallelization to speed up heavy processes.

Input parameters

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

# Globbing pattern common for the samples to be read. In this case,
# files/directories ending in "s" corresponding to the spectra in the dataset:
glob <- "*s"
```

Code to run

```r
# Lister of NMR files within the directory
listed_files <- file_lister(dataset_path_nmr, glob = glob)

# Function for parallelization and speed up the loading (using the number of cores)
plan(multiprocess, workers = num_workers)

# Function to read listed samples from the directory
nmr_dataset <- nmr_read_samples(listed_files)

# Stop parallelization
plan(sequential)
```

Node 2: Append metadata

The metadata can be merged. To do that, an Excel file containing a first column called “NMRExperiments” with the name of the imported spectra is needed.

Input parameters

```r
# The path where metada is contained. Remember that first column needs to be
# called as "NMRExperiment".
excel_file <- "C:/Users/fmadrid/MTBLS242/nmr_dataset_metadata_tidy.xlsx"
```

Code to run

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

Node 3: Interpolation

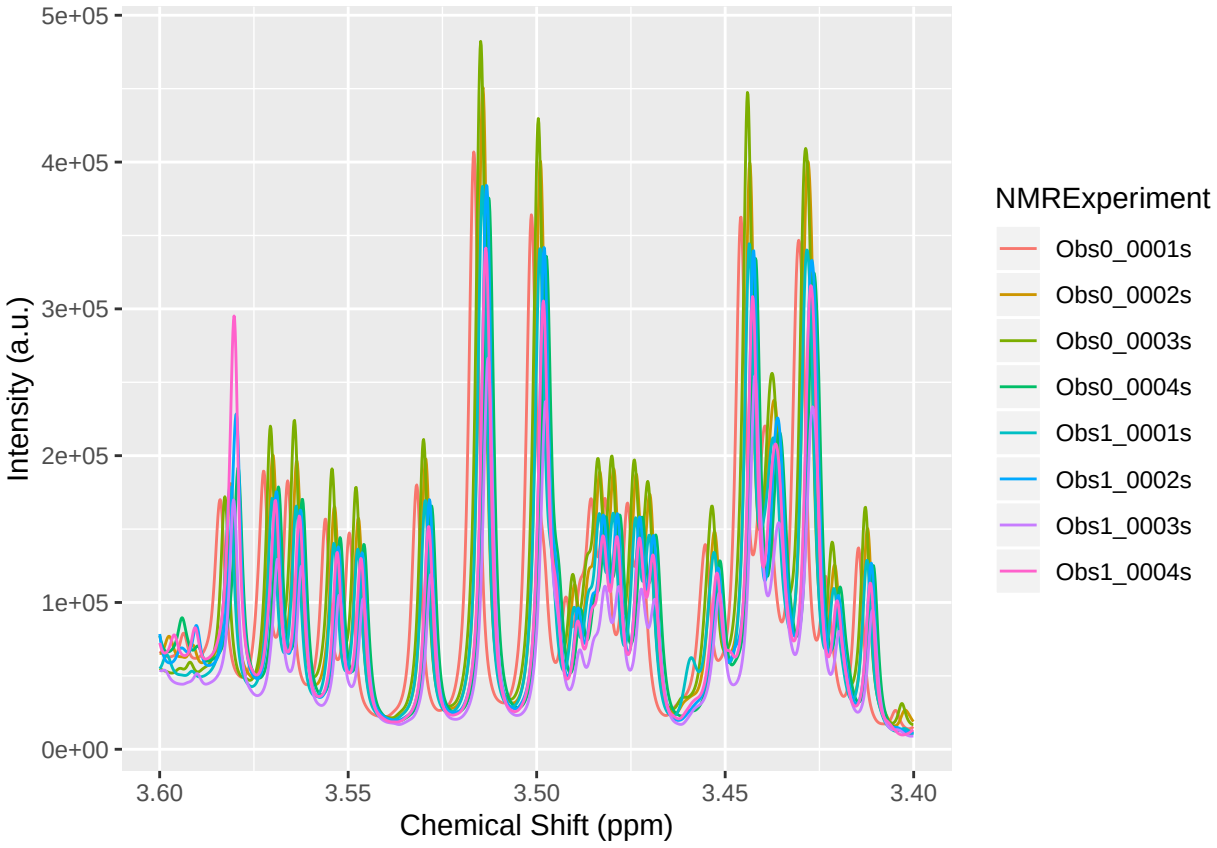
Interpolation is used to unify the ppm axis of all spectra. However, a ppm range can also be set which would limit regions containing noise. The ppm resolution is automatically calculated with the function nmr_ppm_resolution in "Code to run". However, ppm resolution can also be set by using the function `ppm_resolution(nmr_dataset)`.

Input parameters

```
ppm_range_start <- 0.7
ppm_range_end <- 9.5
```

Code to run

```
# Build the ppm axis with the minimum ppm value (max), the maximum ppm value
# (min) and the resolution or distance between datapoints (by)

# You can leave by = NULL or calculate with ppm_resolution(nmr_dataset)
axis <- c(min = ppm_range_start, max = ppm_range_end, by = NULL)

# Interpolation
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(3.60, 3.40))
```



Node 4: Region Exclusion

Solvent regions and / or other potential chemical artefacts can be removed from the spectra by removing specific ppm regions. In this case, the biological samples (serum) and solvent contained water and its signal should be removed. To do this, a vector containing the range (min ppm value, max ppm value) of the signal(s) to be removed are defined, for example: `exclude_regions <- list(water = c(4.5, 5.1), methanol = c(3.33, 3.34))`.

Input parameters

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

Code to run

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

Node 5: Initial Outlier Rejection

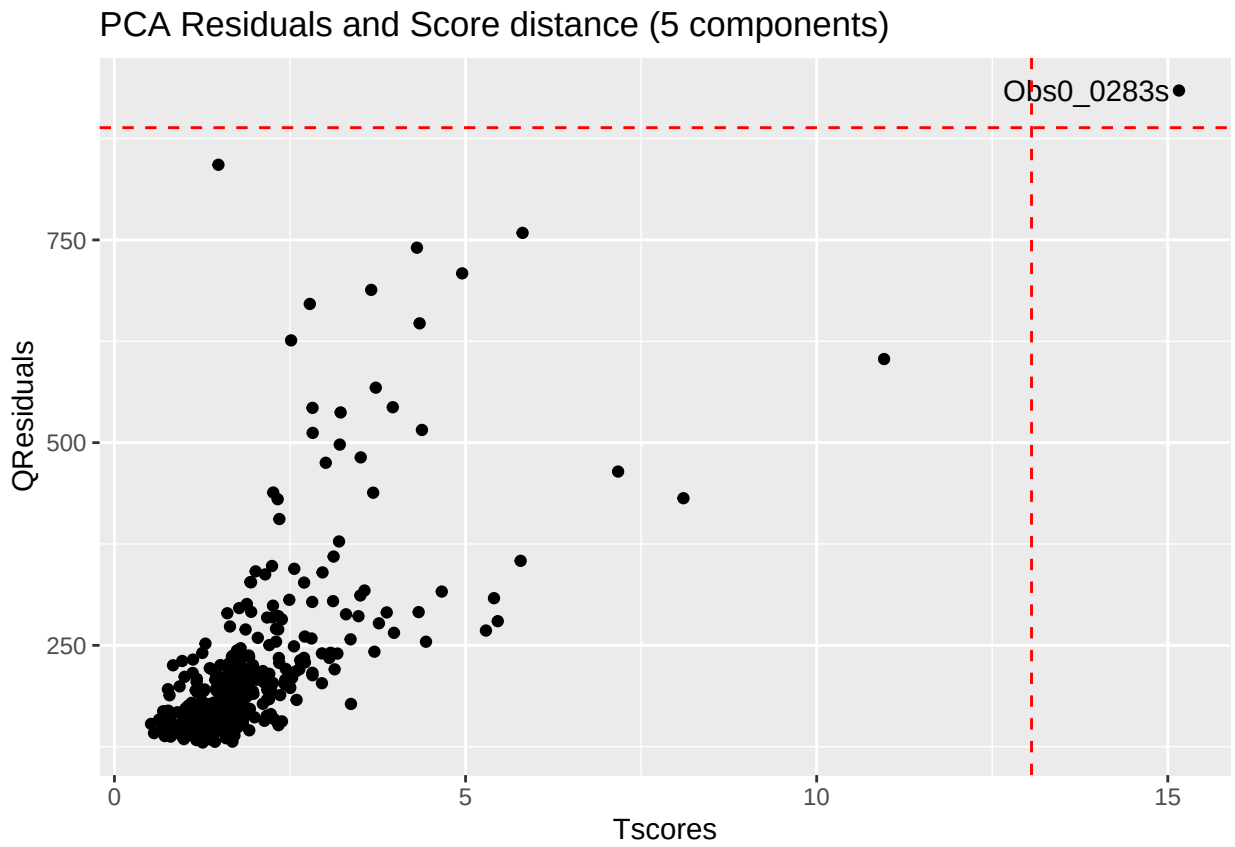
The robust principal component analysis (rPCA) identifies for outliers in the dataset. The standard threshold, based on quantiles for Q residual and T2 score values, results in less sensitivity to extreme intensities. The exclusion of samples can be then defined. The plot below indicates that the sample "Obs0_0283s" differs greatly on QResiduals and Tscores from other samples (top-right corner). Outlier samples are annotated as part of the pipeline and not used for downstream analysis.

Input parameters

```
# No input parameters are needed in this section
```

Code to run

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



Specifically, this sample is discarded by running the function below or can manually be ignored.

```
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 certain samples. In this case, we compare two levels of the variable Timepoint in the MTBLS242 dataset: 'preop' and '3 months after surgery'. However, other conditions can be 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

The `filter` function can be then used on the `nmr_dataset` to selected `Timepoint` levels: "preop" and "3 months after surgery". The operator `%in%` can be used to select determined elements within the Timepoint vector.

Code to run

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

Node 7: Peak detection and Alignment

Peak detection is based on a combination of factors: 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 <- 3

# Maximum alignment shift
maxShift_ppm <- 0.0015
```

Code to run

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

# For parallelization
plan(multiprocess, workers = num_workers)

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

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

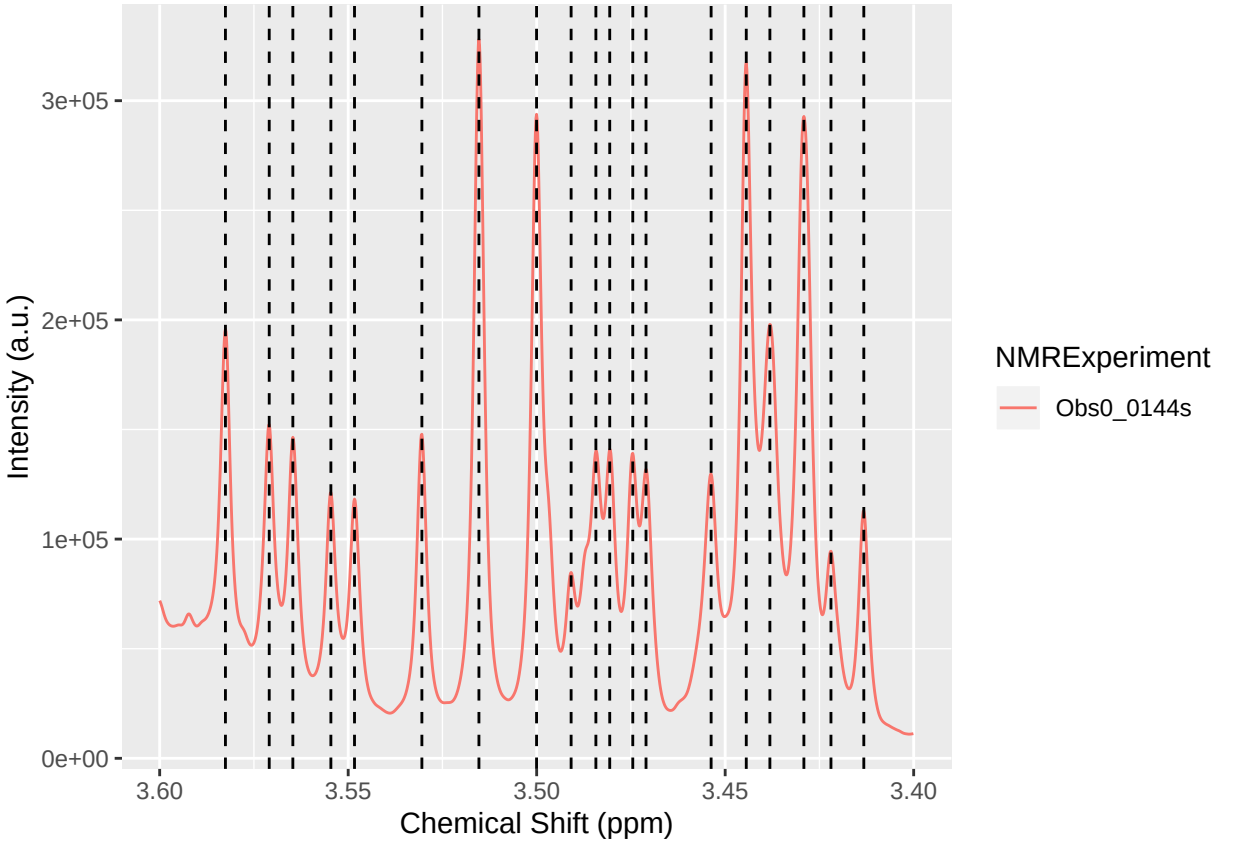
```r
peak_data <- nmr_detect_peaks(nmr_dataset,
                              nDivRange_ppm = nDivRange_ppm,
                              scales = scales,
                              baselineThresh = baselineThresh,
                              SNR.Th = SNR.Th)

# 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...
```

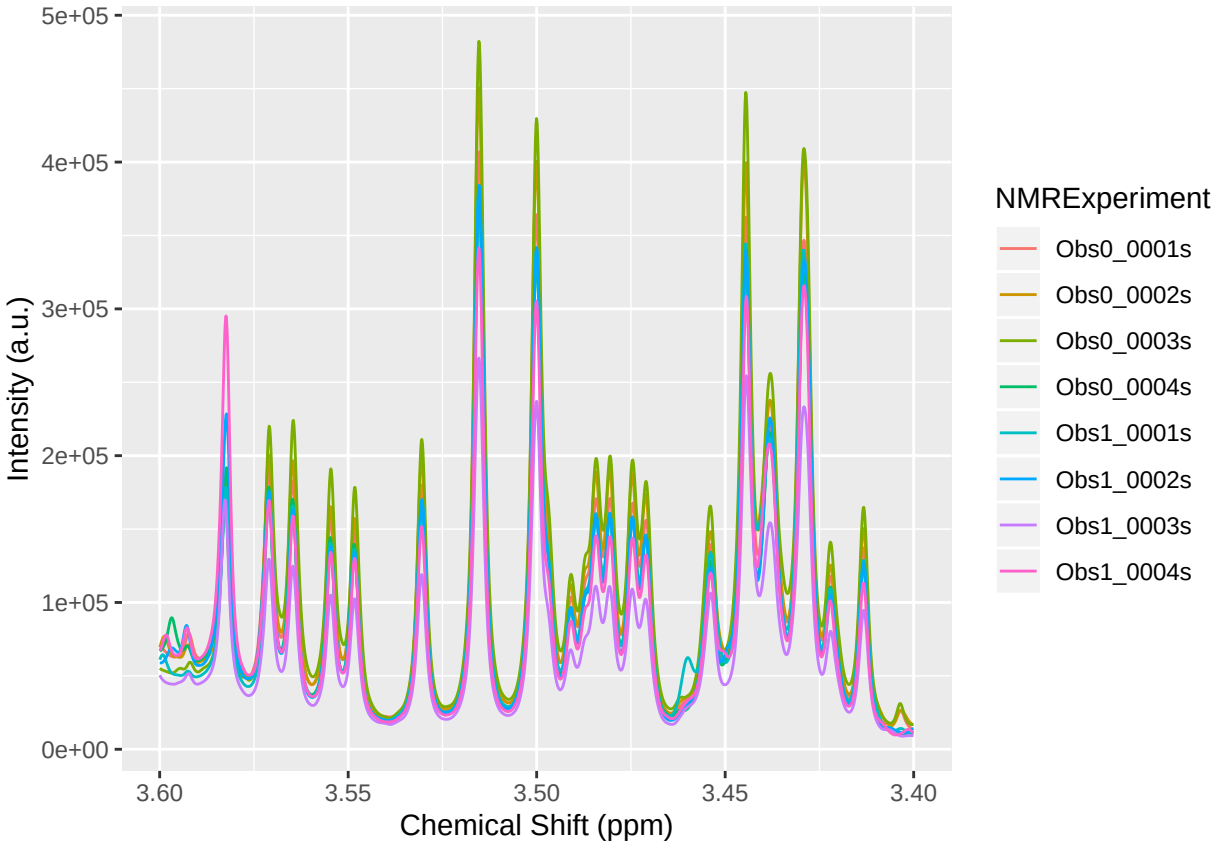
```r
nmr_dataset <- nmr_align(nmr_dataset, peak_data,
                         NMRExp_ref = NMRExp_ref,
                         maxShift_ppm = maxShift_ppm,
                         acceptLostPeak = acceptLostPeak)

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




```
# Stop parallelization
plan(sequential)
```

The peaks detected can be further inspected. If the argument `interactive= TRUE` in the function `plot(nmr_dataset, interactive = TRUE)`, an interactive plot will be generated allowing to be zoomed in. It can be saved in HTML files as `plot_webgl(nmr_dataset, html_filename = "your_path/plot.html")`.

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



Node 8: Normalization

The dataset can be normalised. This is recommended for biological samples, controlling for dilution factors, irregular sample preparation, etc. Probabilistic quotient normalization (PQN) is one of the most used model-based techniques in NMR-based metabolomics.

Additionally, the function `nmr_normalize_extra_info` allows getting diagnostic information on the normalization process, by extracting the normalization factors applied to each spectrum individually compared to the median. The function also creates a diagnostic plot of all normalization factors in relation to the median.

Input parameters

```
# No input parameters are needed in this section
```

Code to run

```
# Probabilistic quotient normalization
nmr_dataset <- nmr_normalize(nmr_dataset, method = "pqn")

# Check the normalization factors (for checking purposes)
```

```
norm_pqn_diagnostic <- nmr_normalize_extra_info(nmr_dataset)
gplt_norm_factor_pqn <- norm_pqn_diagnostic$plot
```

Node 9: Integration

Peak integration is calculated using peak width, done automatically with set `peak_width_ppm = NULL` or manually, in which users can select a specific peak width for integrating the detected peaks. Note: peaks other than those from the reference spectrum can be added automatically (see help). This AlpsNMR step differs from the bucketing approach in which spectra are equally divided into buckets (for example in intervals of 0.01 ppm) that lead into a higher number of total variables. The bucketing approach has the inconvenience of peaks being split into several parts, lowering the statistical power, and vice-versa (certain overlapping peak tails might result in false positives).

Input parameters

```
peak_width_ppm <- NULL
```

Code to run

```
# Set the peak list of the reference sample to get the positions of peaks.
peak_list_ref <- filter(peak_data, NMRExperiment == NMRExp_ref)

# Integrate those peaks positions
nmr_peak_table <- nmr_integrate_peak_positions(

  # In all samples in the nmr_dataset
  samples = nmr_dataset,

  # Integrate those positions. You can introduce a ppm numeric vector or a
  # dataframe with a "ppm" variable
  peak_pos_ppm = peak_list_ref,

  # With this width peaks. If NULL, this is calculated automatically
  peak_width_ppm = peak_width_ppm)
```

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

## peak_pos_ppm input introduced as dataframe
```

```
# Get a final version of the processed dataset (metadata + integrated peaks)
nmr_peak_table_completed <- get_integration_with_metadata(nmr_peak_table)
```

Node 10: Machine learning

A pairwise multilevel approach takes into consideration variability within the same individual. In this case, the metabolomic profile of two time-points of the same individuals (before and 3 months after surgery). The

function `rdCV_PLS_RF_ML` performs a multilevel repeated double cross-validation optimized for unbiased variable selection (MUVR algorithm, see Shi et al., 2018). The double cross-validation procedure comprises an inner "tuning" loop nested within an outer loop aimed at reducing bias resulting from overfitting models to experimental data. Then, autoselected variables are ranked according to their significance in VIP values.

Modelling through the multivariate modelling with minimally biased variable selection (MUVR) algorithm

```
model <- rdCV_PLS_RF(nmr_data(nmr_peak_table),
                     Y = nmr_peak_table_completed$Timepoint)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

##
## Missing ID -> Assume all unique (i.e. sample independence)
## Y is factor -> Classification (2 classes)

## Setting levels: control = FALSE, case = TRUE

## Setting direction: controls < cases

## Setting levels: control = FALSE, case = TRUE

## Setting direction: controls < cases

## Setting levels: control = FALSE, case = TRUE

## Setting direction: controls < cases

## Setting levels: control = FALSE, case = TRUE

## Setting direction: controls < cases

## Setting levels: control = FALSE, case = TRUE

## Setting direction: controls < cases

## Setting levels: control = FALSE, case = TRUE

## Setting direction: controls < cases

##
##  Elapsed time 0.07266667 mins
```

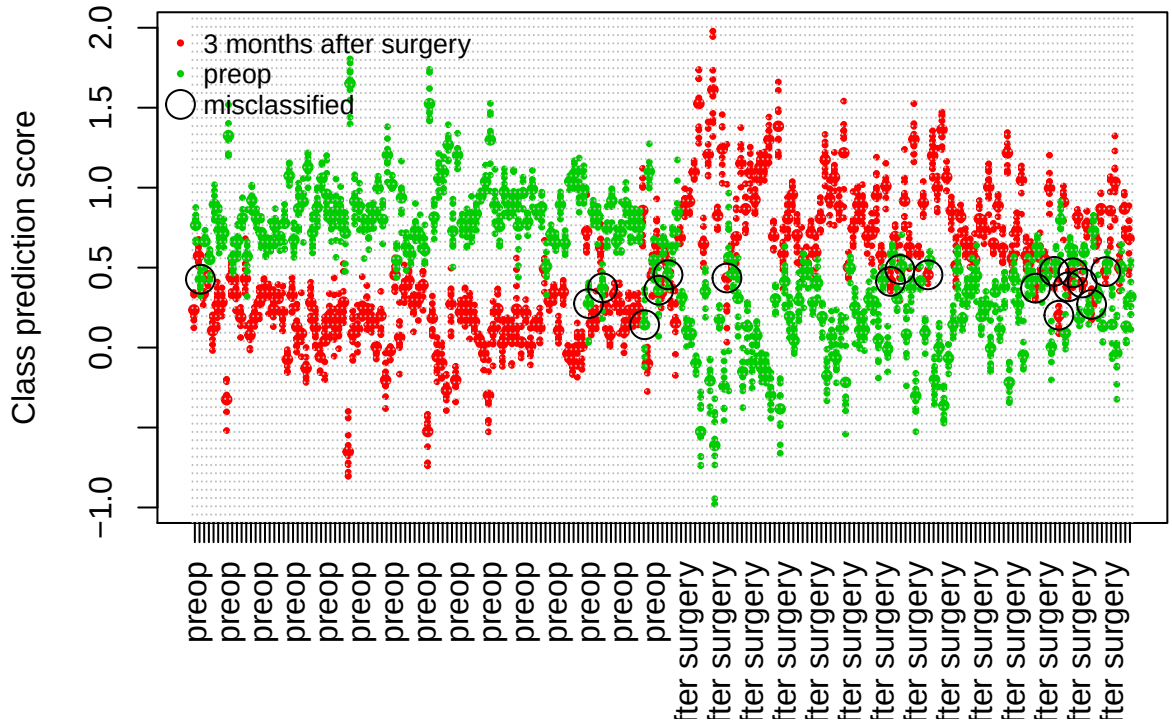
```
AUC_model(model)
```

```
## AUC model is 0.968774781919112
```

The figure below shows the "misclassification plot" from a multilevel PLS-DA model between `preop` and `3 months after surgery`. The misclassification plot shows each sample predicted as `preop` and `3 months after surgery`. The first half of the horizontal axis represents actual `preop` samples, while the second half represents actual `3 months after surgery` samples. Bold points are the average predicted samples, while grey points display predictions from each iteration. Misclassifications are differences between 3-month outcomes predicted from baseline compared to actual 3-month values and are shown by the black circles. To generate a misclassification plot which gives the information about the actual class and predicted class, the following function can be used:

Model performance

```
MUVR_model_plot(model)
```



Permutation test

Permutation tests can be performed to test the statistical significance of the models. The function `permutation_test_model` randomly assigns the labels of the samples to build a new model in each iteration. The number of permutations (iterations) can be set with `nPerm`. For convenience, we have

adjusted the number of permutations to 20 due to the extended computing time; but for a better result, a minimum of 100 permutations should be considered (Szymanska et al., 2012).

```
permutations = permutation_test_model(model, nPerm = 20)
```

```
##
## "MVObj" permutation 1 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.07066667 mins
##
## Estimated time left: 1.342667 mins
##
##
## "MVObj" permutation 2 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.06733333 mins
##
## Estimated time left: 1.242 mins
##
##
## "MVObj" permutation 3 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.05816667 mins
##
## Estimated time left: 1.111611 mins
##
##
## "MVObj" permutation 4 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.05933333 mins
##
## Estimated time left: 1.022 mins
##
##
## "MVObj" permutation 5 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.05966667 mins
##
## Estimated time left: 0.9455 mins
##
##
## "MVObj" permutation 6 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.06283333 mins
##
## Estimated time left: 0.882 mins
##
##
```

```
## "MVObj" permutation 7 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.0585 mins
##
## Estimated time left: 0.8106429 mins
##
##
## "MVObj" permutation 8 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.05966667 mins
##
## Estimated time left: 0.74425 mins
##
##
## "MVObj" permutation 9 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.06533333 mins
##
## Estimated time left: 0.6866852 mins
##
##
## "MVObj" permutation 10 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.058 mins
##
## Estimated time left: 0.6198333 mins
##
##
## "MVObj" permutation 11 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.05833333 mins
##
## Estimated time left: 0.5548636 mins
##
##
## "MVObj" permutation 12 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.061 mins
##
## Estimated time left: 0.4927778 mins
##
##
## "MVObj" permutation 13 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.06083333 mins
##
## Estimated time left: 0.4307692 mins
```

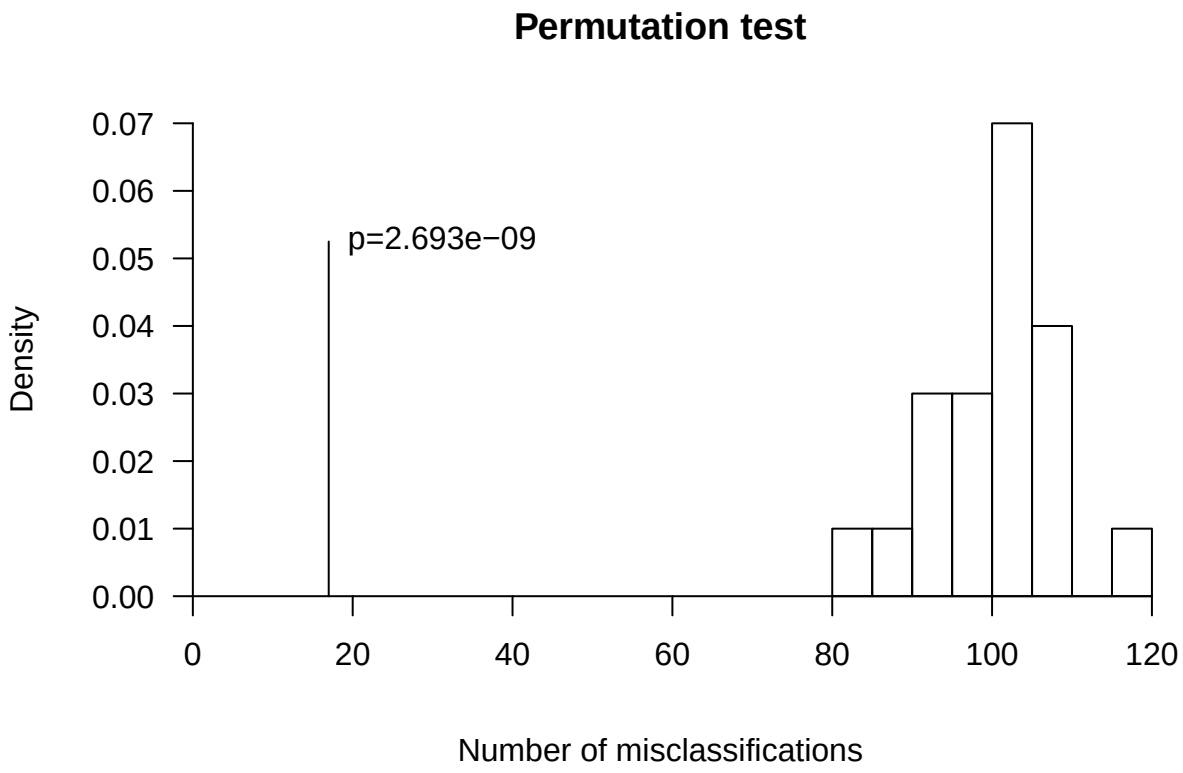
```
##
##
## "MVObj" permutation 14 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.05916667 mins
##
## Estimated time left: 0.3682143 mins
##
##
## "MVObj" permutation 15 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.05983333 mins
##
## Estimated time left: 0.3063333 mins
##
##
## "MVObj" permutation 16 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.06966667 mins
##
## Estimated time left: 0.2471667 mins
##
##
## "MVObj" permutation 17 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.063 mins
##
## Estimated time left: 0.1855882 mins
##
##
## "MVObj" permutation 18 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.06016667 mins
##
## Estimated time left: 0.123537 mins
##
##
## "MVObj" permutation 19 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.06033333 mins
##
## Estimated time left: 0.06169298 mins
##
##
## "MVObj" permutation 20 of 20
##
## Y is factor -> Classification (2 classes)
##  Elapsed time 0.06116667 mins
```

```
##
## Estimated time left: 0 mins
```

Permutation test plot

Permutation test plot for multilevel PLS-DA model gives a p-value for the model performance. The actual number of misclassifications departs from the null hypothesis distribution.

```
permutation_test_plot(model, permutations)
```



Autoselected features

Then, the most significant metabolic features, VIP values, from the autoselected features can be extracted.

```
VIPs= model_VIP(model)
VIPs
```

```
##            order       name    rank
## ppm_3.5826     1 ppm_3.5826   2.440
## ppm_2.2481     2 ppm_2.2481   3.100
## ppm_1.2141     3 ppm_1.2141   5.375
## ppm_2.7471     4 ppm_2.7471   6.925
## ppm_1.3538     5 ppm_1.3538   7.995
```

```
## ppm_3.0626     6 ppm_3.0626   8.005
## ppm_4.1376     7 ppm_4.1376   8.370
## ppm_4.1258     8 ppm_4.1258   8.880
## ppm_4.1142     9 ppm_4.1142   9.305
## ppm_1.0663    10 ppm_1.0663   9.450
## ppm_1.0160    11 ppm_1.0160   9.460
## ppm_1.3422    12 ppm_1.3422  11.025
## ppm_1.0546    13 ppm_1.0546  12.030
## ppm_1.0044    14 ppm_1.0044  12.595
## ppm_4.1491    15 ppm_4.1491  17.775
## ppm_1.1857    16 ppm_1.1857  18.515
## ppm_1.1959    17 ppm_1.1959  20.430
## ppm_1.6004    18 ppm_1.6004  21.985
## ppm_0.9904    19 ppm_0.9904  23.305
## ppm_1.2942    20 ppm_1.2942  24.100
## ppm_1.2025    21 ppm_1.2025  25.170
## ppm_2.0653    22 ppm_2.0653  25.735
## ppm_0.8478    23 ppm_0.8478  28.350
## ppm_3.2317    24 ppm_3.2317  29.935
## ppm_2.1543    25 ppm_2.1543  32.265
## ppm_4.0725    26 ppm_4.0725  32.470
## ppm_0.9800    27 ppm_0.9800  32.715
## ppm_0.9693    28 ppm_0.9693  32.895
## ppm_1.5923    29 ppm_1.5923  34.740
## ppm_5.3249    30 ppm_5.3249  36.180
## ppm_1.5813    31 ppm_1.5813  36.740
## ppm_0.9569    32 ppm_0.9569  36.770
## ppm_5.3119    33 ppm_5.3119  38.075
## ppm_1.3215    34 ppm_1.3215  39.630
## ppm_3.3822    35 ppm_3.3822  48.665
## ppm_3.2280    36 ppm_3.2280  49.980
## ppm_0.9091    37 ppm_0.9091  51.600
## ppm_1.5063    38 ppm_1.5063  51.860
## ppm_3.8951    39 ppm_3.8951  53.700
## ppm_2.0594    40 ppm_2.0594  55.260
## ppm_0.8973    41 ppm_0.8973  55.865
## ppm_1.5688    42 ppm_1.5688  55.925
## ppm_2.2362    43 ppm_2.2362  58.080
## ppm_3.2830    44 ppm_3.2830  60.675
## ppm_2.0879    45 ppm_2.0879  63.880
## ppm_0.9447    46 ppm_0.9447  64.945
## ppm_2.0069    47 ppm_2.0069  67.585
## ppm_1.9959    48 ppm_1.9959  69.305
## ppm_2.2631    49 ppm_2.2631  70.015
## ppm_1.4942    50 ppm_1.4942  71.620
## ppm_5.2978    51 ppm_5.2978  71.825
## ppm_5.3522    52 ppm_5.3522  75.665
## ppm_3.2696    53 ppm_3.2696  81.600
## ppm_3.4710    54 ppm_3.4710  81.630
## ppm_2.7651    55 ppm_2.7651  82.085
## ppm_2.0193    56 ppm_2.0193  83.630
## ppm_3.8145    57 ppm_3.8145  86.400
## ppm_3.4843    58 ppm_3.4843  86.435
## ppm_3.6837    59 ppm_3.6837  91.075
```

```
## ppm_3.4806    60 ppm_3.4806  93.785
## ppm_1.9848    61 ppm_1.9848  95.075
## ppm_0.8851    62 ppm_0.8851  95.105
## ppm_2.0971    63 ppm_2.0971  98.025
## ppm_3.5000    64 ppm_3.5000  98.060
## ppm_3.4132    65 ppm_3.4132  99.490
## ppm_3.5153    66 ppm_3.5153 101.025
## ppm_3.2543    67 ppm_3.2543 101.165
## ppm_3.4291    68 ppm_3.4291 101.680
```

Node 11: Identification

Finally, AlpsNMR includes an identification step that uses the source of the biological sample (plasma/serum, urine and cells) generating a ranked dataframe with ppm and proposed candidates in the Human Metabolome Database (http://www.hmdb.ca). However, given the signal overlap between several potential compounds at a given ppm region, a user needs to verify the results. A vector with significant ppm values is extracted and analyzed by the function `nmr_identify_regions_blood`. The number of proposed candidates can be set. In this particular case, 3 were set. Some of the identified metabolites were automatically found by this process, while others (not assigned, `NA`) could still be identified (see Supplementary Table 1 in Supplementary Material).

Autoselected features (blood samples)

```
# Put ppm values into a numeric vector
ppm_to_assign <- ppm_VIP_vector(VIPs)

# Identify ppm values in the ppm vector
assignation <- nmr_identify_regions_blood(ppm_to_assign,
                                          num_proposed_compounds = 3)

# Cleaner version without NAs
assignation_NArm <- na.omit(assignation)

# Display the most useful columns
assignation_NArm[,-6:-8]
```

```
##                          Metabolite HMDB_code Shift_ppm Type
## 7    2-Hydroxy-3-methylbutyric acid HMDB00407     0.835    d
## 17       3-Methyl-2-oxovaleric acid HMDB00491     0.915    t
## 88                  Isovaleric acid HMDB00718     0.915    d
## 921                     Ketoleucine HMDB00695     0.945    d
## 92                      Ketoleucine HMDB00695     0.945    d
## 101  2-Hydroxy-3-methylbutyric acid HMDB00407     0.965    d
## 10   2-Hydroxy-3-methylbutyric acid HMDB00407     0.965    d
## 1782                       L-Valine HMDB00883     0.991    d
## 1781                       L-Valine HMDB00883     0.991    d
## 1031     L-Alpha-aminobutyric acid HMDB00452     0.997    t
## 178                        L-Valine HMDB00883     0.991    d
## 103      L-Alpha-aminobutyric acid HMDB00452     0.997    t
## 179                        L-Valine HMDB00883     1.044    d
```

```
## 70                          Ethanol HMDB00108     1.185    t
## 701                         Ethanol HMDB00108     1.185    t
## 84                  Isobutyric acid HMDB01873     1.225    d
## 1411                  L-Lactic acid HMDB00190     1.335    d
## 141                   L-Lactic acid HMDB00190     1.335    d
## 1011                      L-Alanine HMDB00161     1.485    d
## 1611                      L-Proline HMDB00162     2.005    m
## 161                       L-Proline HMDB00162     2.005    m
## 1612                      L-Proline HMDB00162     2.005    m
## 9    2-Hydroxy-3-methylbutyric acid HMDB00407     2.025    m
## 301                   Acetylglycine HMDB00532     2.065    s
## 891                 Isovaleric acid HMDB00718     2.065    d
## 30                    Acetylglycine HMDB00532     2.065    s
## 89                  Isovaleric acid HMDB00718     2.065    d
## 163                       L-Proline HMDB00162     2.075    m
## 91                     Ketoleucine HMDB00695     2.105    m
## 127                    L-Glutamine HMDB00641     2.140    m
## 97               L-Acetylcarnitine HMDB00201     2.145    s
## 281                         Acetone HMDB01659     2.235    s
## 28                          Acetone HMDB01659     2.235    s
## 180                       L-Valine HMDB00883     2.276    m
## 187                      Ornithine HMDB00214     3.061    t
## 107                    L-Arginine HMDB00517     3.245    t
## 133                    L-Histidine HMDB00177     3.245   dd
## 381                         Betaine HMDB00043     3.265    s
## 261               3-Methylhistidine HMDB00479     3.255    m
## 38                          Betaine HMDB00043     3.265    s
## 26                3-Methylhistidine HMDB00479     3.255    m
## 164                       L-Proline HMDB00162     3.425   dt
## 1641                      L-Proline HMDB00162     3.425   dt
## 207                         Sucrose HMDB00258     3.475    t
## 2072                        Sucrose HMDB00258     3.475    t
## 2071                        Sucrose HMDB00258     3.475    t
## 41                          Choline HMDB00097     3.522   dd
## 1                 1-Methylhistidine HMDB00001     3.695    s
## 211                         Sucrose HMDB00258     3.685    s
## 217                         Uridine HMDB00296     3.816   dd
## 37                          Betaine HMDB00043     3.905    s
## 46                     Creatinine HMDB00562     4.065    s
## 40                          Choline HMDB00097     4.071  ddd
## 1401                  L-Lactic acid HMDB00190     4.115    q
## 140                   L-Lactic acid HMDB00190     4.115    q
## 1621                      L-Proline HMDB00162     4.135   dd
## 2221                        Uridine HMDB00296     4.136    m
## 162                       L-Proline HMDB00162     4.135   dd
## 222                         Uridine HMDB00296     4.136    m
## 1622                      L-Proline HMDB00162     4.135   dd
## 2222                        Uridine HMDB00296     4.136    m
##                J_Hz ppm_to_assign
## 7              6.87        0.8478
## 17             7.47        0.9091
## 88             6.61        0.9091
## 921            6.68        0.9447
## 92             6.68        0.9569
```

```
## 101             6.96        0.9569
## 10              6.96        0.9693
## 1782            7.01        0.9800
## 1781            7.01        0.9904
## 1031            7.58        0.9904
## 178             7.01        1.0044
## 103             7.58        1.0044
## 179             7.05        1.0546
## 70              7.08        1.1857
## 701             7.08        1.1959
## 84              7.02        1.2141
## 1411            6.96        1.3215
## 141             6.96        1.3422
## 1011            7.28        1.4942
## 1611                        1.9959
## 161                         2.0069
## 1612                        2.0193
## 9    13.81 6.88 3.75        2.0193
## 301                         2.0594
## 891             0.53        2.0594
## 30                          2.0653
## 89              0.53        2.0653
## 163                         2.0879
## 91                          2.0971
## 127                         2.1543
## 97                          2.1543
## 281                         2.2362
## 28                          2.2481
## 180  14.03 7.01 4.41        2.2631
## 187                         3.0626
## 107             6.93        3.2317
## 133       16.10 4.93        3.2317
## 381                         3.2543
## 261                         3.2543
## 38                          3.2696
## 26                          3.2696
## 164       11.65 7.02        3.4132
## 1641      11.65 7.02        3.4291
## 207              9.3        3.4710
## 2072             9.3        3.4806
## 2071             9.3        3.4843
## 41       5.816 4.162        3.5153
## 1                           3.6837
## 211                         3.6837
## 217       12.77 4.49        3.8145
## 37                          3.8951
## 46                          4.0725
## 40                          4.0725
## 1401       6.93 6.93        4.1142
## 140        6.93 6.93        4.1258
## 1621       8.63 6.42        4.1258
## 2221                        4.1258
## 162        8.63 6.42        4.1376
## 222                         4.1376
```

```
## 1622       8.63 6.42        4.1491
## 2222                        4.1491
```

REFERENCES

Shi,L. et al. (2018) Variable selection and validation in multivariate modelling. Bioinformatics.

Szymańska,E. et al. (2012) Double-check: validation of diagnostic statistics for PLS-DA models in metabolomics studies. Metabolomics, 8, 3–16.