Introduction to AlpsNMR

AlpsNMR authors

2024-04-30

Abstract

An introduction to the AlpsNMR package, showing the most relevant functions and a proposed workflow. This includes loading bruker NMR samples, adding sample annotations, preprocessing the spectra, detecting outliers, detecting peaks, aligning the samples and integrating the peaks to build a peak table.

Package

AlpsNMR 4.5.0

Contents

1 Getting started . . . 3
2 Enable parallellization . . . 3
3 Data: The `MeOH_plasma_extraction` dataset . . . 3
4 Loading samples . . . 4
5 Adding annotations . . . 5
6 Interpolation . . . 5
7 Plotting samples . . . 5
8 Exclude regions . . . 6
9 Filter samples . . . 6
10 Robust PCA for outlier detection . . . 6
11 Baseline estimation . . . 7
12 Peak detection . . . 9
13 Spectra alignment . . . 12
14 Normalization . . . 13

15 Peak grouping. 15
16 Session Info: 22

1 Getting started

The `AlpsNMR` package has most of its functions prefixed with `nmr_`. The main reason for this is to avoid conflicts with other packages. Besides, it helps for autocompletion: Most coding environments such as RStudio will let you see most of the function names by typing `nmr_` followed by pressing the tab key.

This vignette assumes some basic knowledge of NMR and data analysis, and some basic R programming.

We will start by loading `AlpsNMR` along some convenience packages:

```
library(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
library(ggplot2)
library(readxl)
library(BiocParallel)
library(AlpsNMR)
## Loading required package: future
##
## Attaching package: 'AlpsNMR'
## The following object is masked from 'package:stats':
##
##     filter
```

2 Enable parallellization

This package is able to parallellize several functions through the use of the `BiocParallel` package. Whether to parallelize or not is left to the user that can control the parallellization registering backends. Please check `vignette("Introduction_To_BiocParallel", package = "BiocParallel")`.

```
#register(SerialParam(), default = TRUE)  # disable parallellization
register(SnowParam(workers = 3, exportglobals = FALSE), default = TRUE)  # enable parallellization with 3 wo
```

3 Data: The `MeOH_plasma_extraction` dataset

To explore the basics of the AlpsNMR package, we have included three NMR samples acquired in a 600 MHz Bruker instrument bundled with the package. The samples are pooled quality control plasma samples, that were extracted with methanol. They only contain small molecules.

If you have installed this package, you can obtain the directory where the samples are with the command:

```
MeOH_plasma_extraction_dir <- system.file("dataset-demo", package = "AlpsNMR")
MeOH_plasma_extraction_dir
## [1] "/private/tmp/Rtmp1VFItu/Rinst5f9b475a9e15/AlpsNMR/dataset-demo"
```

The demo directory includes three zipped Bruker samples and a dummy Excel metadata file:

```
list.files(MeOH_plasma_extraction_dir)
## [1] "10.zip"              "20.zip"              "30.zip"
## [4] "README.txt"          "dummy_metadata.xlsx"
```

Since these are quality control samples, the metadata is a dummy table:

```
MeOH_plasma_extraction_xlsx <- file.path(MeOH_plasma_extraction_dir, "dummy_metadata.xlsx")
annotations <- readxl::read_excel(MeOH_plasma_extraction_xlsx)
annotations
## # A tibble: 3 x 3
##   NMRExperiment SubjectID TimePoint
##   <chr>         <chr>     <chr>
## 1 10            Ana       baseline
## 2 20            Ana       3 months
## 3 30            Elia      baseline
```

4 Loading samples

The function to read samples is called `nmr_read_samples()`. It expects a character vector with the samples to load that can be paths to directories of Bruker format samples or paths to JDX files.

Additionally, this function can filter by pulse sequences (e.g. load only NOESY samples) or loading only metadata.

```
zip_files <- fs::dir_ls(MeOH_plasma_extraction_dir, glob = "*.zip")
zip_files
## /private/tmp/Rtmp1VFItu/Rinst5f9b475a9e15/AlpsNMR/dataset-demo/10.zip
## /private/tmp/Rtmp1VFItu/Rinst5f9b475a9e15/AlpsNMR/dataset-demo/20.zip
## /private/tmp/Rtmp1VFItu/Rinst5f9b475a9e15/AlpsNMR/dataset-demo/30.zip
dataset <- nmr_read_samples(sample_names = zip_files)
dataset
## An nmr_dataset (3 samples)
```

If your samples happen to be in different **folders per class**, AlpsNMR provides convenience functions to read them as well. With this example:

```
- your_dataset/
  + control/
     * 10/
     * 20/
     * 30/
  + mutated/
     * 10/
     * 20/
```

```
    * 30/
```

You could use:

```
dataset <- nmr_read_samples_dir(c("your_dataset/control", "your_dataset/mutated"))
dataset
```

If after reading the `?nmr_read_samples` page you still have issues, feel free to open an issue at https://github.com/sipss/AlpsNMR/issues and ask for clarification.

5 Adding annotations

We can embed the external annotations we loaded above into the dataset:

```
dataset <- nmr_meta_add(dataset, metadata = annotations, by = "NMRExperiment")
```

And retrieve them from the dataset:

```
nmr_meta_get(dataset, groups = "external")
## # A tibble: 3 x 3
##   NMRExperiment SubjectID TimePoint
##   <chr>         <chr>     <chr>
## 1 10            Ana       baseline
## 2 20            Ana       3 months
## 3 30            Elia      baseline
```

If you want to learn more about sample metadata (including acquisition and FID processing parameters), as well as more complex ways of adding annotations, check out the `vignette("Vig02-handling-metadata-and-annotations", package = "AlpsNMR")`.

6 Interpolation

1D NMR samples can be interpolated together, in order to arrange all the spectra into a matrix, with one row per sample. Here we choose the range of ppm values that we want to include in further analyses.

```
dataset <- nmr_interpolate_1D(dataset, axis = c(min = -0.5, max = 10))
```

If the `axis = NULL` then the ppm axis is autodetected from the samples.

See `nmr_interpolate_1D()` for further reference on the axis options.

7 Plotting samples

Plotting many spectra with so many points is quite expensive so it is possible to include only some regions of the spectra or plot only some samples.

```
plot(dataset, NMRExperiment = c("10", "30"), chemshift_range = c(2.2, 2.8))
```

8 Exclude regions

Some regions can easily be excluded from the spectra with `nmr_exclude_region()`:

```
regions_to_exclude <- list(water = c(4.6, 5), methanol = c(3.33, 3.39))
dataset <- nmr_exclude_region(dataset, exclude = regions_to_exclude)
plot(dataset, chemshift_range = c(4.2, 5.5))
```

9 Filter samples

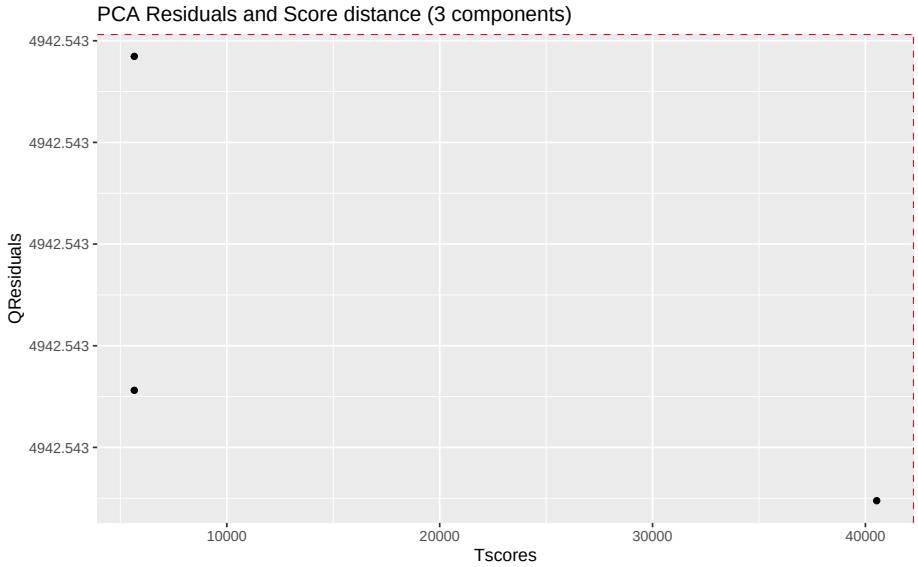
Maybe we just want to analyze a subset of the data, e.g., only a class group or a particular gender. We can filter some samples according to their metadata as follows:

```
samples_10_20 <- filter(dataset, SubjectID == "Ana")
nmr_meta_get(samples_10_20, groups = "external")
## # A tibble: 2 x 3
##   NMRExperiment SubjectID TimePoint
##   <chr>         <chr>     <chr>
## 1 10            Ana       baseline
## 2 20            Ana       3 months
```

10 Robust PCA for outlier detection

The AlpsNMR package includes robust PCA analysis for outlier detection.

```
pca_outliers_rob <- nmr_pca_outliers_robust(dataset, ncomp = 3)
nmr_pca_outliers_plot(dataset, pca_outliers_rob)
```



Samples with greater QResiduals and Tscores than the threshold defined by the red line are candidates for further exploration and exclusion. With this small dataset, there is not much to see.

11 Baseline estimation

Spectra may display an unstable baseline, specially when processing blood/fecal samples.

The peak detection and integration algorithms benefit from having an estimation of the baseline, so it is advisable to compute it first and check it fits as expected.

See before:

```
plot(dataset, chemshift_range = c(1.37, 2.5))
```

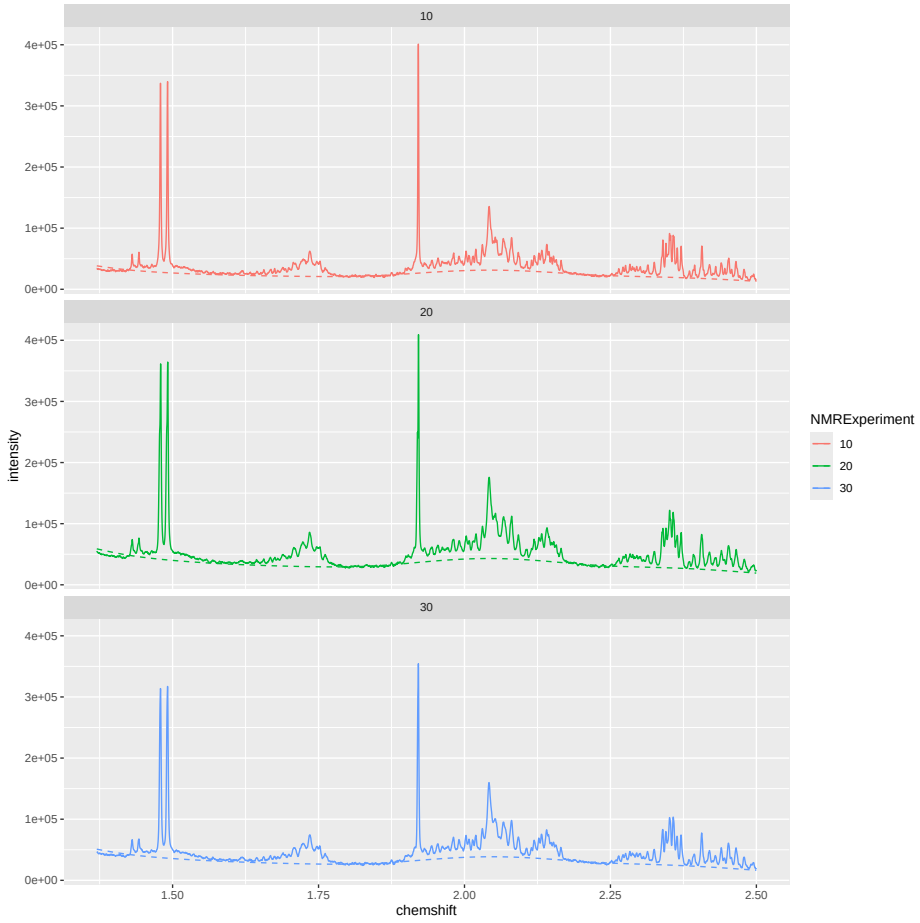
```
plot(dataset, chemshift_range = c(3.5,3.8))
```

Estimate the baseline:

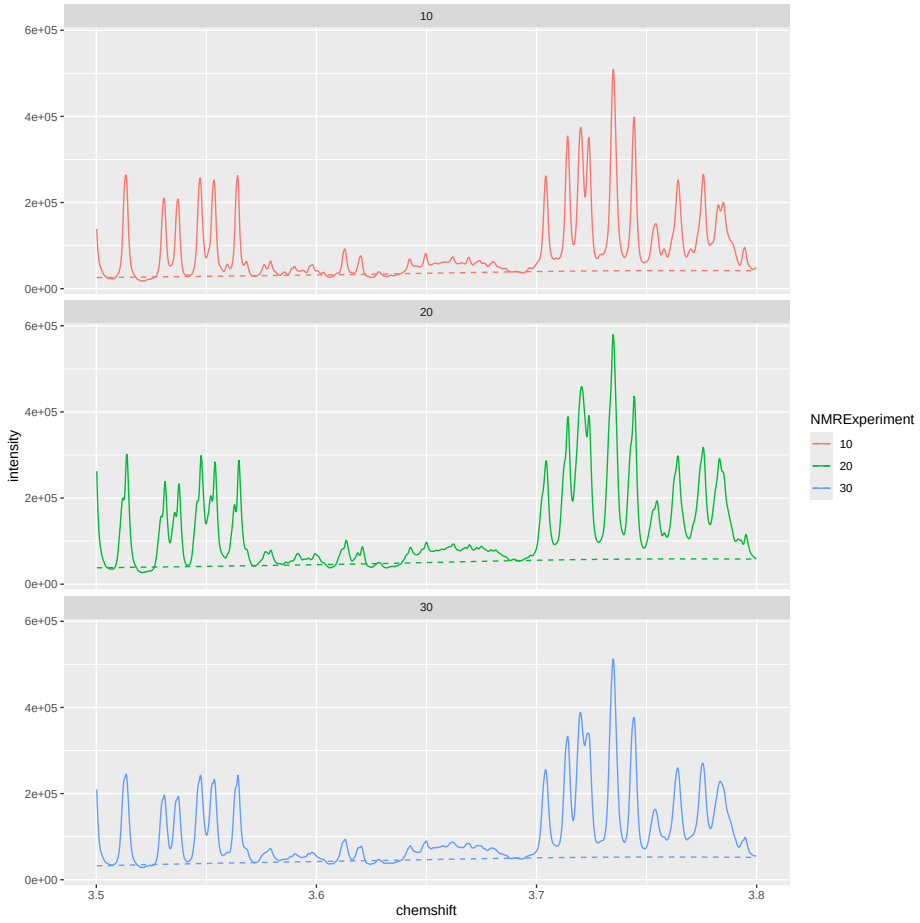
```
dataset <- nmr_baseline_estimation(dataset, lambda = 9, p = 0.01)
```

And after:

```
# TODO: Simplify this plot
spectra_to_plot <- tidy(dataset, chemshift_range = c(1.37, 2.5))
baseline_to_plot <- tidy(dataset, chemshift_range = c(1.37, 2.5), matrix_name = "data_1r_baseline")

ggplot(mapping = aes(x = chemshift, y = intensity, color = NMRExperiment)) +
    geom_line(data = spectra_to_plot) +
    geom_line(data = baseline_to_plot, linetype = "dashed") +
    facet_wrap(~NMRExperiment, ncol = 1)
```




```
# TODO: Simplify this plot
spectra_to_plot <- tidy(dataset, chemshift_range = c(3.5, 3.8))
baseline_to_plot <- tidy(dataset, chemshift_range = c(3.5, 3.8), matrix_name = "data_1r_baseline")

ggplot(mapping = aes(x = chemshift, y = intensity, color = NMRExperiment)) +
    geom_line(data = spectra_to_plot) +
    geom_line(data = baseline_to_plot, linetype = "dashed") +
    facet_wrap(~NMRExperiment, ncol = 1)
```



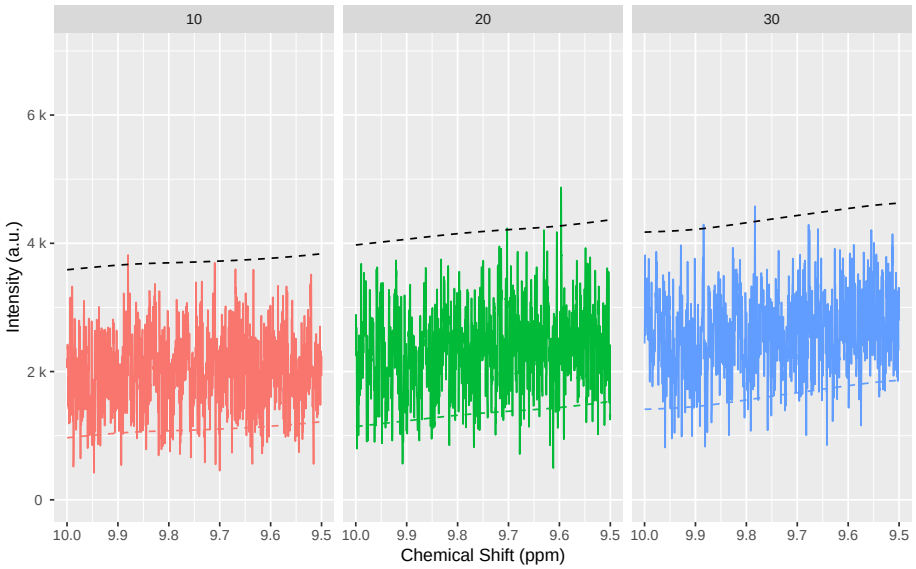
12 Peak detection

The peak detection is performed on short spectra segments using a continuous wavelet transform. Peaks below a threshold intensity are automatically discarded.

Our current approach relies on the use of the baseline threshold (`baselineThresh`) automatically calculated (see `?nmr_baseline_threshold`) and the Signal to Noise Threshold (`SNR.Th`) to discriminate valid peaks from noise.

See `?nmr_detect_peaks` for more information.

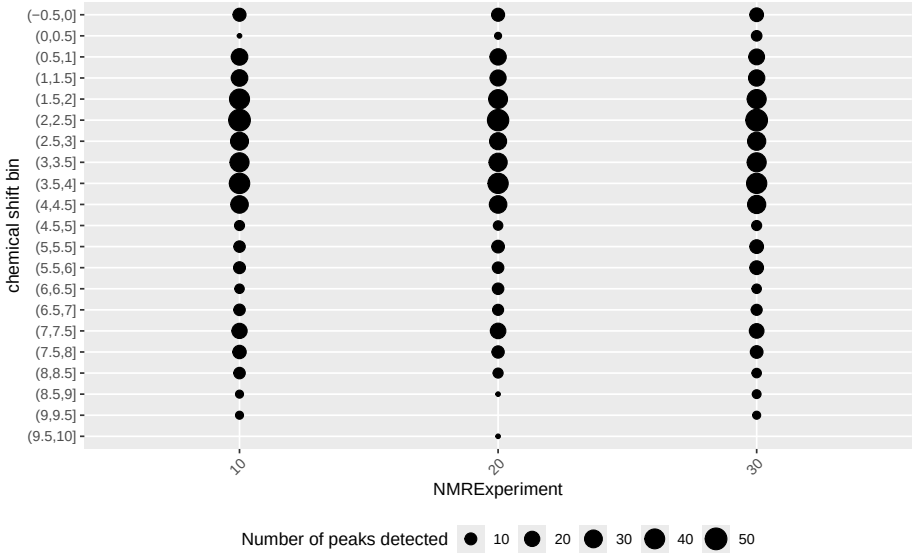
```
baselineThresh <- nmr_baseline_threshold(dataset, range_without_peaks = c(9.5, 10), method = "median3mad")
nmr_baseline_threshold_plot(dataset, baselineThresh)
```




```
peak_list_initial <- nmr_detect_peaks(
    dataset,
    nDivRange_ppm = 0.1,
    scales = seq(1, 16, 2),
    baselineThresh = baselineThresh,
    SNR.Th = 3,
    fit_lorentzians = TRUE
)
```

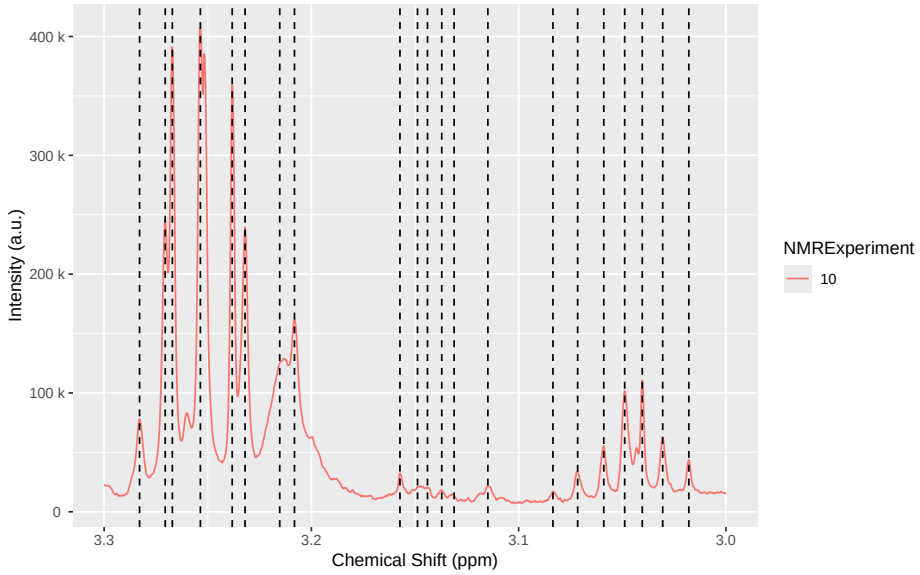
We can get an overview of the number of peaks we detect on each sample and each chemical shift region:

```
nmr_detect_peaks_plot_overview(peak_list_initial)
```



We can explore in a more detailed way the detected peaks:

```
nmr_detect_peaks_plot(dataset, peak_list_initial, NMRExperiment = "10", chemshift_range = c(3, 3.3))
```

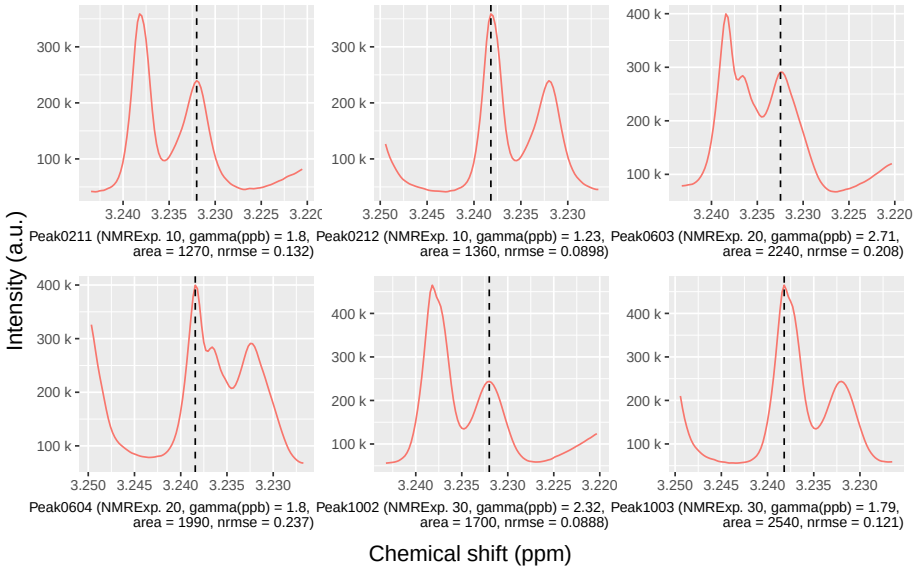


Let's the detected peaks in a smaller region across samples:

```
peak_list_in_range <- filter(peak_list_initial, ppm > 3.22, ppm < 3.24)
peak_list_in_range
## # A tibble: 6 x 11
##   peak_id  NMRExperiment   ppm   pos intensity_raw intensity ppm_infl_min
##   <chr>    <chr>         <dbl> <dbl>         <dbl>     <dbl>        <dbl>
## 1 Peak0211 10             3.23 16281       239459.   224420.         3.23
## 2 Peak0212 10             3.24 16308       358753.   343534.         3.24
## 3 Peak0603 20             3.23 16283       291094.   267693.         3.23
## 4 Peak0604 20             3.24 16309       399656.   375897.         3.24
## 5 Peak1002 30             3.23 16281       243670.   232273.         3.23
## 6 Peak1003 30             3.24 16308       464835.   453429.         3.24
## # i 4 more variables: ppm_infl_max <dbl>, gamma_ppb <dbl>, area <dbl>,
## #   norm_rmse <dbl>
```

```
plot(dataset, chemshift_range = c(3.22, 3.25))
```

```
nmr_detect_peaks_plot_peaks(
    dataset,
    peak_list_initial,
    peak_ids = peak_list_in_range$peak_id,
    caption = paste("{peak_id}",
                    "(NMRExp.\u00A0{NMRExperiment},",
                    "gamma(ppb)\u00a0=\u00a0{gamma_ppb},",
                    "\narea\u00a0=\u00a0{area},",
                    "nrmse\u00a0=\u00a0{norm_rmse})")
)
```




```
peak_list_initial_accepted <- peaklist_accept_peaks(
    peak_list_initial,
    dataset,
    area_min = 50,
    keep_rejected = FALSE,
    verbose = TRUE
)
## Acceptance report
## i 833/1127 peaks accepted. (73.9%)
## i Removing 294 peaks
```

13 Spectra alignment

Once we have a preliminary peak list, we can align the spectra using the `nmr_align()` function. We expect shifts between the spectra, this becomes necessary so we can cluster the peaks correctly afterwards and build a peak table.

The alignment process takes several parameters, including:

- `NMRExp_ref`: An NMRExperiment with a reference sample. Usually it should be a pool of all samples if it is available. Otherwise, you can use `nmr_align_find_ref()` to find a sample. Depending on how heterogeneous your dataset is, there may not be a good reference sample (even if the function picks one, the alignment might not succeed), so please always check the results afterwards.
- `maxShift_ppm`: The maximum shift allowed when aligning the spectra.
- `acceptLostPeak`: Set it to `TRUE` if you want to accept some peaks getting lost during the alignment process. Since the peak detection is never perfect, it is reasonable to accept some lost peaks.

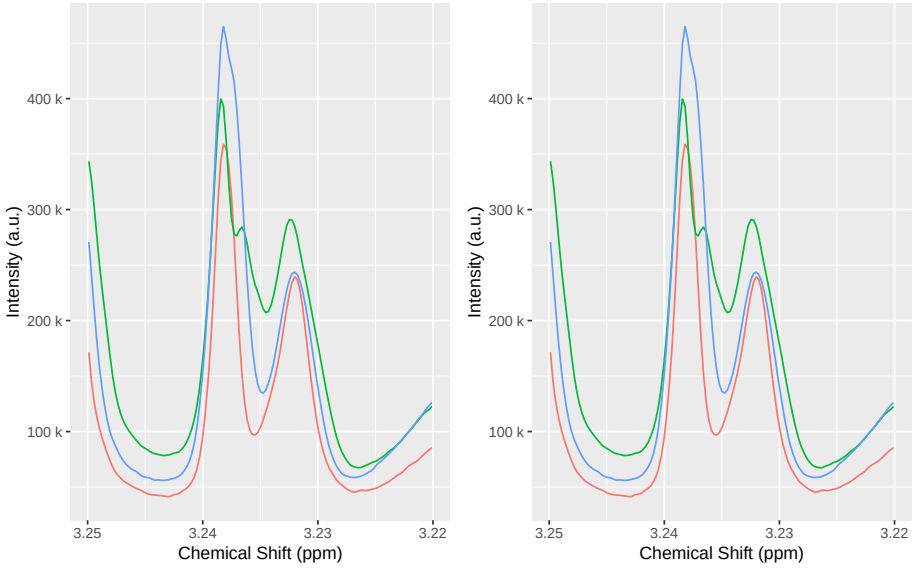
```
NMRExp_ref <- nmr_align_find_ref(dataset, peak_list_initial_accepted)
message("Your reference is NMRExperiment ", NMRExp_ref)
## Your reference is NMRExperiment 20
```

```
dataset_align <- nmr_align(
    nmr_dataset = dataset,
    peak_data = peak_list_initial_accepted,
    NMRExp_ref = NMRExp_ref,
    maxShift_ppm = 0.0015,
    acceptLostPeak = TRUE
)
```

Compare the dataset before and after alignment, to verify the quality of the alignment:

```
plot(dataset, chemshift_range = c(3.025, 3.063))
plot(dataset_align, chemshift_range = c(3.025, 3.063))
```

```
cowplot::plot_grid(
    plot(dataset, chemshift_range = c(3.22, 3.25)) + theme(legend.position = "none"),
    plot(dataset_align, chemshift_range = c(3.22, 3.25)) + theme(legend.position = "none")
)
```



14 Normalization

With the spectra correctly aligned, you can use spectra normalization techniques. We normalize after alignment because some of the normalization techniques are sensitive to misalignments.

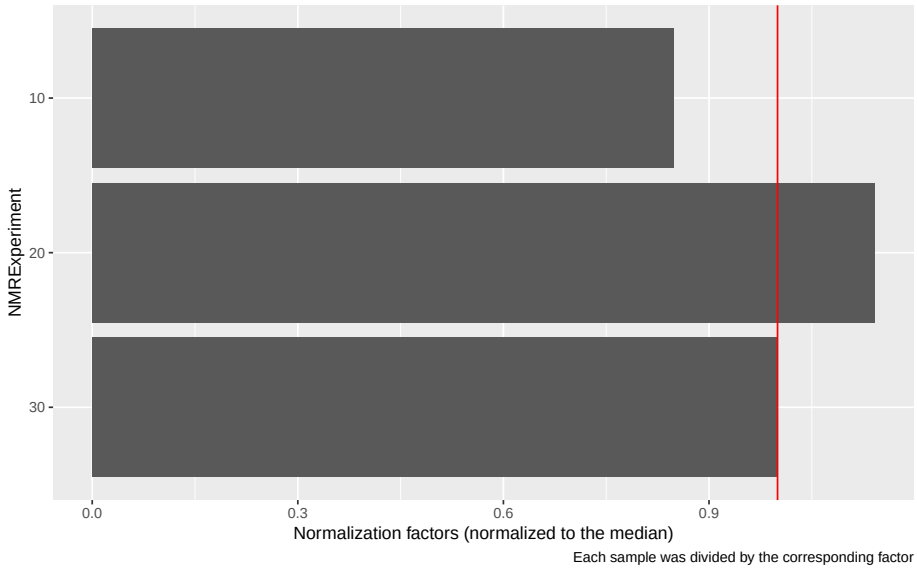
There are multiple normalization techniques available. The most strongly recommended is the Probabilistic Quantile Normalization (`pqn`), but it requires more samples for its internal estimations to be reliable, as it needs a computation of the median spectra. Nevertheless, it is possible to compute it:

```
dataset_norm <- nmr_normalize(dataset_align, method = "pqn")
## Warning: There are not enough samples for reliably estimating the median spectra
## i The Probabalistic Quotient Normalization requires several samples to compute the median spectra. Your nu
## i Review your peaks before and after normalization to ensure there are no big distortions
```

The normalization essentially computes a normalization factor for each sample.

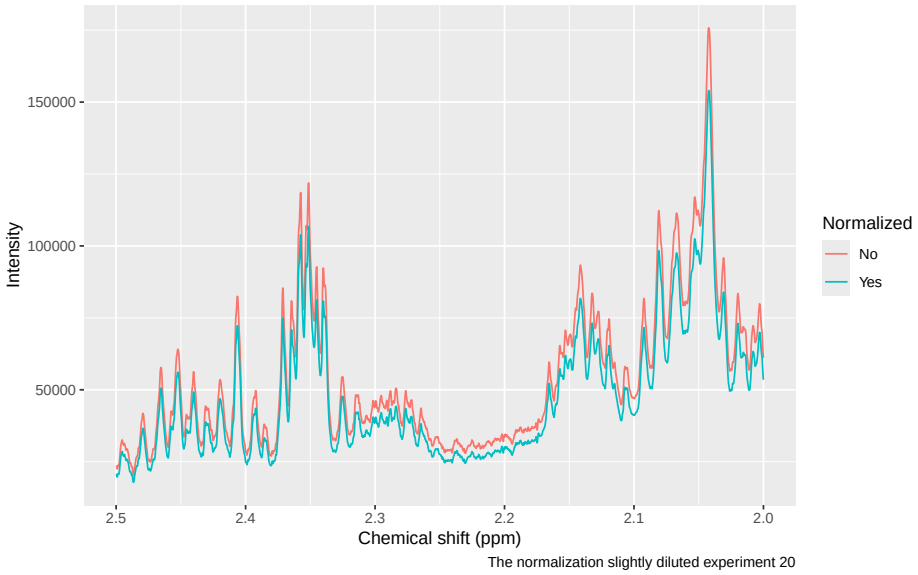
The plot shows the dispersion with respect to the median of the normalization factors, and can highlight samples with abnormally large or small normalization factors.

```
normalization_info <- nmr_normalize_extra_info(dataset_norm)
normalization_info$norm_factor
##   NMRExperiment norm_factor norm_factor_norm
## 1            10   0.8478185        0.8478185
## 2            20   1.1417458        1.1417458
## 3            30   1.0000000        1.0000000
normalization_info$plot
```



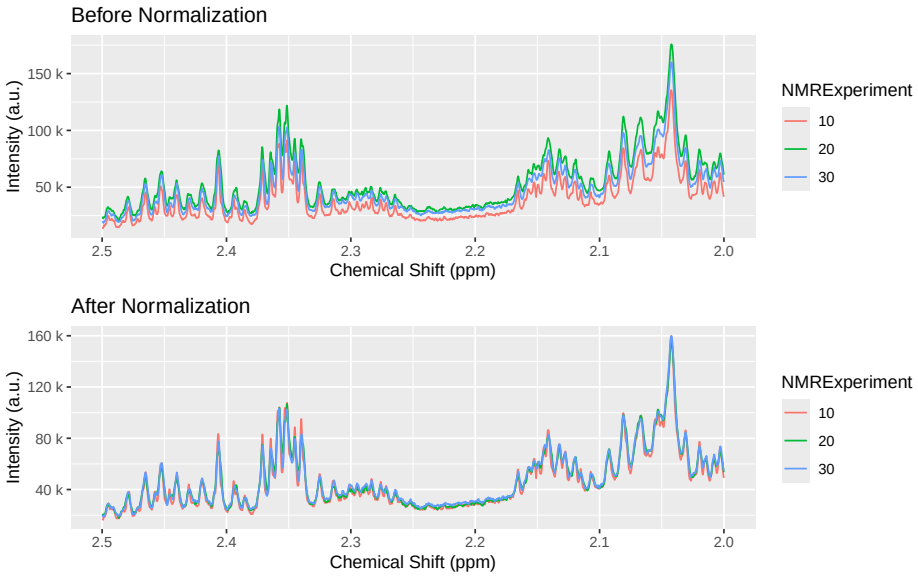
We can confirm sample 20 is now slightly more diluted:

```
to_plot <- dplyr::bind_rows(
    tidy(dataset_align, NMRExperiment = "20", chemshift_range = c(2,2.5)) %>%
        mutate(Normalized = "No"),
    tidy(dataset_norm, NMRExperiment = "20", chemshift_range = c(2,2.5)) %>%
        mutate(Normalized = "Yes"),
)
ggplot(data = to_plot, mapping = aes(x = chemshift, y = intensity, color = Normalized)) +
    geom_line() +
    scale_x_reverse() +
    labs(y = "Intensity", x = "Chemical shift (ppm)",
         caption = "The normalization slightly diluted experiment 20")
```



And all samples are more homogeneous now:

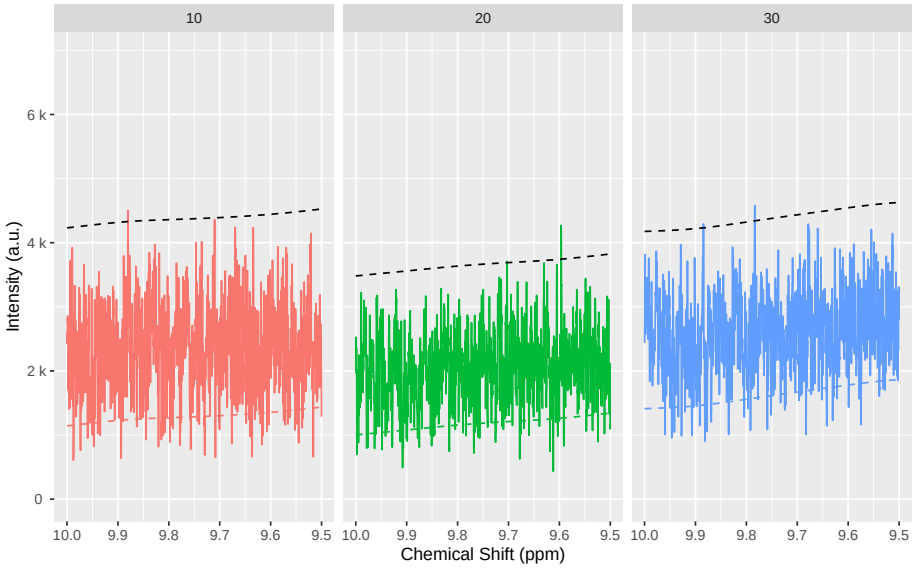
```
cowplot::plot_grid(
    plot(dataset_align, chemshift_range = c(2, 2.5)) + labs(title="Before Normalization"),
    plot(dataset_norm, chemshift_range = c(2, 2.5)) + labs(title="After Normalization"),
    ncol = 1
)
```



15 Peak grouping

If you align or normalize your samples, you should rerun the peak detection to ensure the peak positions and estimations are well calculated:

```
baselineThresh <- nmr_baseline_threshold(dataset_norm, range_without_peaks = c(9.5, 10), method = "median3mad")
nmr_baseline_threshold_plot(dataset_norm, baselineThresh)
```



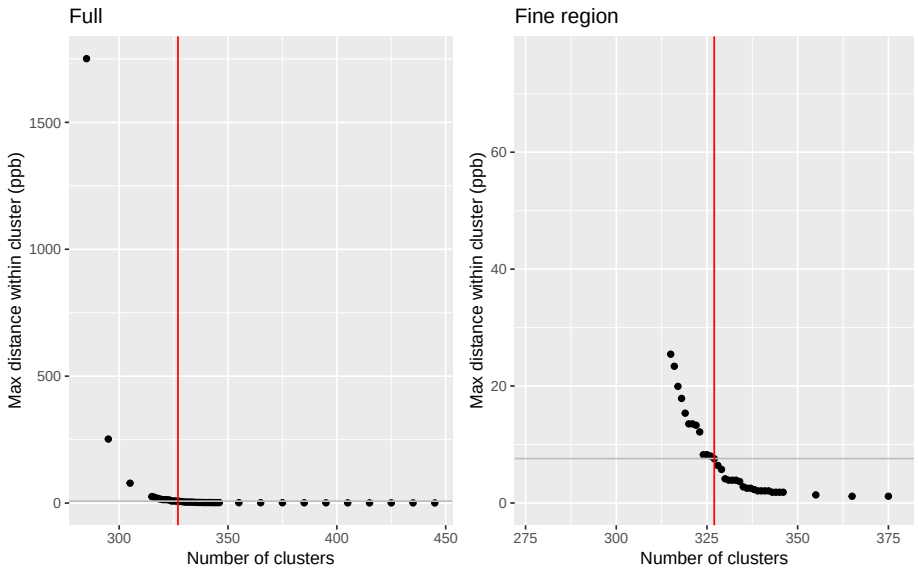

```
peak_list_for_clustering_unfiltered <- nmr_detect_peaks(
    dataset_norm,
    nDivRange_ppm = 0.1,
    scales = seq(1, 16, 2),
    baselineThresh = baselineThresh,
    SNR.Th = 3,
    fit_lorentzians = TRUE,
    verbose = TRUE
)

peak_list_for_clustering <- peaklist_accept_peaks(
    peak_list_for_clustering_unfiltered,
    dataset_norm,
    area_min = 50,
    keep_rejected = FALSE,
    verbose = TRUE
)
## Acceptance report
## i 837/1128 peaks accepted. (74.2%)
## i Removing 291 peaks
```

Feel free to plot, explore and further curate your peak list. Or proceed with the current one:

Once we have a peak list for each sample `peak_list`, we need to turn it into a table, merging peaks from different samples together.

```
clustering <- nmr_peak_clustering(peak_list_for_clustering, verbose = TRUE)
## i The maximum distance between two peaks in the same cluster is of 7.6 ppbs
```

```
cowplot::plot_grid(
    clustering$num_cluster_estimation$plot + labs(title = "Full"),
    clustering$num_cluster_estimation$plot +
        xlim(clustering$num_cluster_estimation$num_clusters-50, clustering$num_cluster_estimation$num_cluste
        ylim(0, 10*clustering$num_cluster_estimation$max_dist_thresh_ppb) +
        labs(title = "Fine region")
)
```

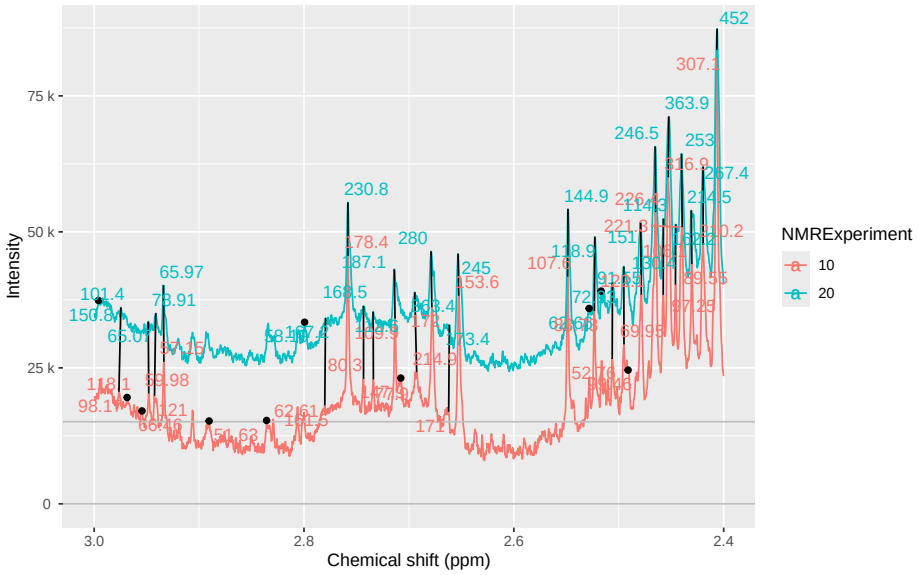



```
peak_list_clustered <- clustering$peak_data
```

We can plot the samples, with the detected peaks and how they have been connected. This allows us to compare the peak detection across samples, and check how good the peak matching is.

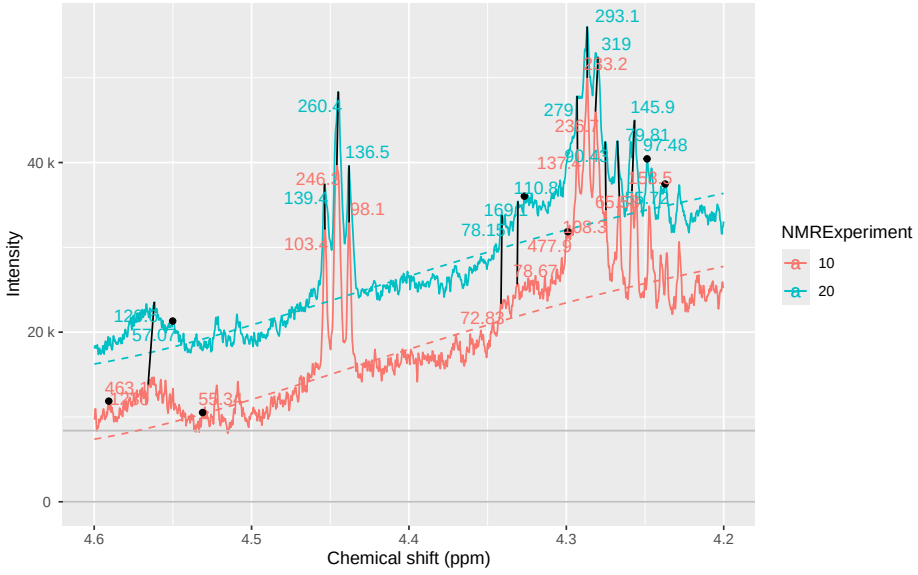
If peaks are matched they are connected with a black segment. If peaks are detected but not matched, they appear as a dot. If you see a peak, without a point on top then it means the peak was not detected or it was filtered out.

```
nmr_peak_clustering_plot(
    dataset = dataset_norm,
    peak_list_clustered = peak_list_clustered,
    NMRExperiments = c("10", "20"),
    chemshift_range = c(2.4, 3.0)
)
```



Sometimes we see peaks and we wonder why are they not detected. We can include the `baselineThresh` to plot it in the sample as well. This can help to diagnose if the `baselineThresh` argument is the cause of a peak not being detected.

```
nmr_peak_clustering_plot(
    dataset_norm,
    peak_list_clustered,
    NMRExperiments = c("10", "20"),
    chemshift_range = c(4.2, 4.6),
    baselineThresh = baselineThresh
)
```




```
peak_table <- nmr_build_peak_table(peak_list_clustered, dataset_norm)
peak_table
## An nmr_dataset_peak_table (3 samples, and 327 peaks)
```

```
peak_matrix <- nmr_data(peak_table)
peak_matrix[1:3, 1:8]
##      0.8119   0.8223   0.8247    0.8333    0.8447   0.8479    0.8553   0.8605
## 10 248.1262       NA       NA  576.0143  583.9754 584.6623  234.5596 316.6228
## 20 411.9895 1448.242 651.2341 1062.1773 1039.3440       NA 1628.4976 426.1570
## 30 286.1167       NA 963.8638  886.4544  859.2279       NA 1761.5355 574.4401
```

Or you can get a data frame with the corresponding annotations:

```
peak_table_df <- as.data.frame(peak_table)
peak_table_df
##    NMRExperiment SubjectID TimePoint   0.8119   0.8223   0.8247    0.8333
## 10            10       Ana  baseline 248.1262       NA       NA  576.0143
## 20            20       Ana  3 months 411.9895 1448.242 651.2341 1062.1773
## 30            30      Elia  baseline 286.1167       NA 963.8638  886.4544
##       0.8447   0.8479    0.8553   0.8605   0.8656   0.8729   0.8784   0.8910
## 10  583.9754 584.6623  234.5596 316.6228 789.1245 293.0063 464.0082 294.2187
## 20 1039.3440       NA 1628.4976 426.1570 754.8024 485.4726 704.8558 622.1860
## 30  859.2279       NA 1761.5355 574.4401 881.7161 544.6914 513.5835 463.6450
##      0.9034   0.9087   0.9158   0.9197   0.9314   0.9437   0.9554    0.9658
## 10 350.8436 341.2886 319.6393 299.8821 299.5249 363.8845 703.9568  937.0603
## 20 719.4185 585.6043       NA 464.9987 456.8719 534.5574 869.8691 1325.2470
## 30 562.9422 539.3870 597.7613 484.1954 447.1457 510.5059 808.2869 1111.3047
##      0.9758    0.9903   1.0020   1.0091   1.0208   1.0409   1.0526    1.0595
## 10 602.7438  752.4888 760.9498 264.6978 228.0143 686.4252 683.5871  84.61342
## 20 843.1016 1010.4784 999.9029 330.4955 295.2787 925.4050 917.6458        NA
## 30 679.3240  940.9196 927.1884 330.1384 273.9946 830.3180 831.3828 126.11232
##       1.0717   1.1749   1.1870   1.1980   1.2085   1.2179   1.2294   1.2366
## 10  57.54959       NA 566.4997 655.1247  913.711 1421.560 1582.320 2743.867
## 20 184.61318 461.9956       NA 929.4433 1477.699 2144.071 3180.656       NA
## 30  82.12987 332.8321 713.8189 915.8183 1364.905 2251.681 2848.689 6061.105
##      1.2408   1.2539   1.2713   1.2807   1.3261   1.3378   1.4309   1.4423
## 10 6760.677 1599.496 459.1637       NA 15599.64 15747.66 108.5837 128.8285
## 20 2881.720 1800.636 766.8679 385.0771 19291.51 19632.59 125.4677 173.6484
## 30 3998.338       NA 712.6686       NA 19158.06 19139.56 126.1861 159.7446
##      1.4792   1.4914   1.5184   1.5228    1.5363    1.5476   1.5587   1.5817
## 10 1269.666 1290.973 196.9007 212.0831  91.68998  59.68399 62.65619 77.91474
## 20 1656.032 1641.459 222.9824 272.9503 125.92324  62.59551       NA       NA
## 30 1532.112 1559.736       NA 185.4979 130.89312 142.13846 54.32585       NA
##      1.6195  1.6319   1.6443   1.6562   1.6677    1.6752    1.6798   1.6885
## 10 106.6812      NA 56.70438 58.18575 125.9570  70.13185  81.70447 186.5148
## 20 130.9576 57.8218 51.23049 62.75341 114.7787 154.95233 118.86423 242.2354
## 30 135.2879      NA 73.51831 82.18996 102.0515 121.55787  87.82803 248.1315
##      1.6929   1.6991   1.7071   1.7099   1.7202    1.7234   1.7349   1.7472
## 10 121.0124 129.5784 226.5209 294.9621 580.2444 1321.8941 653.9540 377.4231
## 20       NA 185.0223 532.7178       NA       NA  828.4124 732.7383 372.6472
## 30 240.6352 172.8209       NA 503.2867       NA  864.8789 667.1569 447.9793
##      1.7525   1.7612   1.7649   1.7782   1.8997   1.9212   1.9315   1.9441
## 10 251.0483 135.7918       NA 62.83582 61.65237 1203.263 315.9743 187.0654
## 20 321.2379 224.2712       NA       NA 78.37190 1330.834 218.9574 331.3299
## 30 265.8839 166.4631 113.3322       NA 75.48978 1514.066 277.9066 227.3127
```

```
##      1.9545   1.9620   1.9721   1.9813   1.9909   2.0031   2.0079   2.0143
## 10 208.9239 176.1407 182.9422 234.5887 224.1196 224.3859 226.4652 208.5121
## 20 299.4998 167.0354 356.3811 353.0796 275.8973 303.7908 237.1840 274.1993
## 30 262.9044 233.6622 230.1438 339.1926 248.8792 273.7877 249.2916 225.4372
##      2.0198   2.0308   2.0423   2.0528    2.0670   2.0711   2.0810   2.0925
## 10 227.3431 321.7810 1302.207 1314.589  752.9154       NA 496.2829 307.7217
## 20 294.0752 419.6870 1508.661 1365.811 1074.0891       NA 695.0037 309.9705
## 30 271.9756 379.8256 1480.583 1221.542  739.5373 747.6153 613.0113 332.4116
##       2.1069    2.1152   2.1195   2.1275   2.1324   2.1413   2.1445   2.1493
## 10  93.78933  76.84992 224.5092 246.5001 319.5974 359.7243 278.7488 207.9623
## 20 147.64953 164.23762 265.0589 298.1992 334.9370 620.8960       NA 451.2703
## 30 114.24108 102.06040 245.9558 240.4358 341.7655 437.8782 406.5889       NA
##      2.1528   2.1573   2.1656   2.2644    2.2717    2.2761    2.2834   2.2878
## 10 201.1596 172.3603 103.8398 60.50906  85.23805  90.78696  96.06216       NA
## 20 256.5145 222.5518 139.4830 75.59151 115.05229 124.85984 149.18578 119.7600
## 30 280.3576 269.9406 128.6531 75.99914 108.31399 116.67778 132.27960 291.0199
##      2.2951   2.3013   2.3141   2.3251   2.3398   2.3451   2.3513   2.3575
## 10 106.1659 127.1724 123.6337 144.0820 335.3851 276.8338 877.0104 706.6683
## 20 165.0579 276.9065 194.2737 190.2546 550.3349 375.2403 813.4507 708.7526
## 30 150.2211 213.4635 167.1972 171.7208 410.4367 354.3635 745.0314 606.8054
##      2.3646   2.3712   2.3852   2.3935   2.4063   2.4196   2.4304   2.4400
## 10 237.0716 244.9379 56.68186 121.2241 307.0505 210.1523 106.0938 221.3220
## 20 393.5357 305.9833 85.19447 183.1241 451.9880 267.4103 214.5316 252.9693
## 30 305.2453 275.6054 72.48182 191.9073 345.6257 236.7438 160.3876 229.3450
##       2.4457   2.4524    2.4574   2.4652   2.4792   2.4911    2.4950   2.5062
## 10  89.54685 316.8724  97.24899 226.3594 122.1125 99.46148  69.95047 52.76363
## 20 162.24949 363.8996 114.25039 246.4821 151.0334       NA 130.40405 72.93065
## 30 142.81027 348.4575 127.04018 294.5953 142.1640 59.25808  84.92035 79.37067
##      2.5161    2.5227   2.5282   2.5484   2.6525   2.6621   2.6782   2.6928
## 10       NA  83.72792       NA 107.5707 153.5688 171.0426 172.0216 214.9457
## 20 91.54535 118.93174 62.65551 144.8924 245.0266 173.3718 279.9865 363.4481
## 30 71.83682  95.15143       NA 127.9623 184.6376 120.0726 194.0600 258.8669
##      2.7073   2.7135    2.7339    2.7433   2.7582   2.7800   2.7994   2.8357
## 10 147.8995 109.8911  77.89777  80.29726 178.3523 161.4813       NA 62.61494
## 20       NA 187.0657 112.56110 168.48956 230.8484 167.2446 58.19063       NA
## 30 223.6738 155.9978 116.89989 100.97569 252.2641 316.3822       NA       NA
##      2.8904   2.9340    2.9413   2.9484   2.9544   2.9627    2.9682    2.9746
## 10 51.63224 57.14955 121.03774 59.97862 66.45715       NA 118.11879  98.16567
## 20       NA 65.96539  78.90826 65.07157       NA       NA        NA 101.44223
## 30       NA 70.23617  79.51778 99.29743       NA 90.71474  84.27172 125.59825
##      2.9939   3.0184   3.0307   3.0406   3.0491   3.0594   3.0718    3.1144
## 10       NA 184.7753 341.1048 415.6746 698.5911 291.5051 164.5374  89.18102
## 20 150.7982 248.2568 460.3224 568.5250 891.9478 417.9656 242.2591 113.06076
## 30 350.8956 234.2364 416.0585 543.3548 769.5021 343.2626 190.0767 105.18072
##      3.1369   3.1444    3.1476    3.1575   3.2082   3.2153   3.2320   3.2382
## 10       NA 65.08866 168.81339  98.41517 1536.217 3769.484 1499.738 1601.331
## 20       NA 61.83531  88.10254 125.50603 1955.028 3815.275 1963.205 1739.073
## 30 50.96765       NA 157.35910 125.98022 1902.083 4290.864 1704.133 2543.183
##      3.2524   3.2673   3.2705   3.2829    3.3019   3.3929    3.4031   3.4083
## 10 4521.605 2016.903 1534.857 462.2252  95.63249  887.957  865.2127 3128.180
## 20 4767.033 2511.020 1702.668 561.0364 110.40016 1067.643        NA 3656.465
```

```
## 30 4419.239 2771.662 1955.109 572.1872 103.75904 1367.236 1421.0174 3495.235
##      3.4186   3.4243   3.4347  3.4445    3.4585   3.4619   3.4681   3.4720
## 10 2274.073 2532.080 1074.992      NA  979.1747 1053.042 1107.150 1199.189
## 20 2702.408 2832.234 1493.641      NA 1423.3779 1045.296 1468.776 1351.868
## 30 2709.228 3323.768 1521.447 264.401 1452.2019 1811.063 1692.697 1945.484
##      3.4833   3.4880   3.4984   3.5137   3.5312   3.5374   3.5477   3.5539
## 10 2848.543 711.1614 3030.239 1153.788 830.7520 804.7678 1107.722 1042.157
## 20 2846.202       NA 3638.346 1235.596 936.8879 847.6890 1232.906 1109.271
## 30 3233.277 874.4151 3395.337 1316.415 972.2546 981.1855 1321.937 1334.910
##       3.5646   3.5768   3.5793   3.5905   3.5986   3.6132   3.6206   3.6428
## 10  968.5097 105.9028 137.8300 131.2571 157.3292 247.7557 176.8860 154.6960
## 20  900.0694       NA 314.2806 123.6738 138.0719 315.5881 135.0540 170.4806
## 30 1187.2639       NA 299.4756 140.5174 229.7466 278.5303 205.2608 189.9662
##      3.6501    3.7042   3.7141   3.7201   3.7237   3.7347   3.7444    3.7540
## 10 208.4098  999.0931 1415.135 2079.832 1549.022 2578.973 1630.252  900.1247
## 20 255.5384 1258.9240 1918.302 3194.135 1693.708 3006.940 2043.422 1140.2085
## 30 278.1656 1159.2643 1639.283 2318.561 2060.678 2769.495 1829.839  880.0379
##      3.7641   3.7755   3.7836   3.7948   3.8234   3.8270   3.8351   3.8438
## 10 1256.336 1298.224 2374.140 236.3243 543.4231 752.9224 3346.587 735.0153
## 20 1596.592 1711.386 2218.180 206.4537 822.1201       NA 4341.257 991.5969
## 30 1399.624 1505.556 2106.734 265.9667 579.1907 819.1823 3585.849 872.1422
##       3.8496   3.8528   3.8567   3.8736   3.8888   3.8924   3.9092   3.9128
## 10  736.9934 1250.343 647.9638 287.8573 1852.677 1898.800 1622.312 1347.827
## 20 1541.0461 1878.056 692.5062 231.6368 2967.161 2273.967 2889.612 1566.574
## 30        NA 1597.724 821.7131 266.5514 2256.631 2431.197 2052.016 1669.685
##      3.9335   3.9493    3.9573   3.9674   3.9741   3.9837   3.9917   4.0020
## 10 216.5390 118.4627  93.12888 205.4455       NA 227.0910 309.6592 413.9531
## 20 268.3245 277.0371 102.80920 141.9612 60.42242 256.5561 306.6140 506.7606
## 30 273.0763 162.3949 103.76530 161.0807 87.12546 230.6409 340.4573 600.5445
##      4.0107   4.0621    4.0976   4.1091   4.1208   4.1322   4.1508   4.2383
## 10 230.8846       NA  958.5921 2880.672 2860.118 1008.417       NA       NA
## 20 228.9913 168.5163 1079.6453 3199.903 3257.002 1245.609       NA 55.71681
## 30 223.0235 181.8002 1126.9300 3367.654 3409.080 1200.469 144.3914 59.04854
##      4.2482   4.2567   4.2670    4.2753   4.2812   4.2867   4.2929   4.2991
## 10       NA 153.4649 65.58097 108.30387 236.6605 233.2040 137.3656 477.9413
## 20 97.47677 145.9449 79.81083  90.43373 319.0322 293.0553 279.0271       NA
## 30 79.35559 129.3181 91.64845 112.55484 267.4956 280.5111 200.6798       NA
##     4.3145   4.3266    4.3308   4.3412    4.4380   4.4454   4.4534   4.5311
## 10      NA       NA  78.66503 72.82589  98.09874 246.2942 103.3528 55.33868
## 20      NA 169.1433 110.84331 78.14891 136.49984 260.4064 139.4042       NA
## 30 84.4525       NA  64.31585       NA 117.17092 228.3749 124.2998       NA
##      4.5533    4.5638   4.5701   4.5754   4.5907   5.0976   5.2369   5.2431
## 10       NA 1229.6755       NA       NA 463.1392       NA 1817.439 1847.273
## 20 57.07267  120.5827       NA       NA       NA       NA 2330.003 2200.440
## 30 58.74334        NA 206.5163 63.23561       NA 126.3221 1957.950 2036.357
##      5.2827   5.2934   5.3025   5.3159   5.3752   5.6172   5.7651  5.7807
## 10       NA 523.6252 234.5118 85.61687       NA 65.78883       NA      NA
## 20 902.7388       NA 441.4213       NA 110.2349       NA 55.28227 82.3699
## 30 445.7218 253.6866       NA 98.27618       NA       NA       NA      NA
##      5.8694   5.8898   5.9157   6.0122    6.0161   6.1030   6.1124   6.1346
## 10       NA       NA       NA       NA 11787.199 116.5275 123.0397       NA
```

```
## 20       NA       NA       NA 4678.138  8102.819 145.0754 129.1509       NA
## 30 65.83419 77.36536 88.34496       NA 12569.618 125.2200 123.6173 53.95125
##      6.9010   6.9154   7.1032    7.1041   7.1088   7.1194   7.1928   7.2038
## 10 122.8229 134.3873       NA        NA 156.3402       NA 189.0228 165.3699
## 20 146.0084 220.0099       NA  59.30034       NA 108.1284 225.4606       NA
## 30 142.6998 161.2664 77.06101 253.88006 247.2531       NA 202.5600       NA
##      7.2068   7.2182   7.2497    7.2772    7.2895   7.3010   7.3281   7.3317
## 10 247.7746 102.8240       NA 110.38726  95.47776 51.04997 119.7981 202.3011
## 20 370.2237 118.3368 62.58484  91.56677 127.29960       NA       NA 340.4834
## 30 348.4245 108.6485       NA  88.71863 143.25901 64.29375       NA 334.5977
##      7.3409  7.3817   7.3941    7.4211   7.4340   7.4462   7.5415   7.5550
## 10 195.5841      NA 55.65824  89.04766 101.8156       NA 74.46895 82.79328
## 20 139.4118      NA 73.21077 102.33101 162.8226 75.26129 69.41431 88.77966
## 30 187.3697 54.1167 59.22346  96.57209 149.4560 75.29813 77.55389 89.19004
##       7.7343   7.7476   7.8720   7.8856    7.9094   7.9182   7.9252  7.9431
## 10 101.52235 89.79723       NA 50.47835 105.81927 143.7226 80.66585      NA
## 20  95.25361 86.37586 54.15299       NA  77.99912       NA       NA 53.8829
## 30  95.99036 80.57030       NA       NA  91.60336 110.0937       NA      NA
##      8.2017   8.2182   8.2202   8.2457   8.3523   8.4610  -0.0003  -0.1483
## 10 157.1990       NA 159.2340 186.2074 234.8404       NA 16859.11 54.34648
## 20 196.4877 208.1011 146.5433 208.4713 222.3893 54.51042 16799.60       NA
## 30 169.1293       NA 185.7654 190.7122 269.4926 55.56135 18104.35       NA
##    -0.1683 -0.1878  -0.2327
## 10      NA      NA       NA
## 20      NA      NA 51.43336
## 30 70.1123 60.5017       NA
```

```
saveRDS(peak_table, "demo_peak_table.rds")
```

From this peak table you can proceed to use statistical testing, machine learning, and any downstream analysis you may be interested in.

16 Session Info:

```
sessionInfo()
## R version 4.4.0 Patched (2024-04-24 r86482)
## Platform: aarch64-apple-darwin20
## Running under: macOS Ventura 13.6.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  LAPACK version
##
## locale:
## [1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/New_York
## tzcode source: internal
##
```

```
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] AlpsNMR_4.5.0       future_1.33.2       BiocParallel_1.37.1
## [4] readxl_1.4.3        ggplot2_3.5.1       dplyr_1.1.4
## [7] BiocStyle_2.31.0
##
## loaded via a namespace (and not attached):
##   [1] baseline_1.3-5         gridExtra_2.3          rlang_1.1.3
##   [4] magrittr_2.0.3         MassSpecWavelet_1.69.0 matrixStats_1.3.0
##   [7] compiler_4.4.0         vctrs_0.6.5            reshape2_1.4.4
##  [10] RcppZiggurat_0.1.6     quadprog_1.5-8         rvest_1.0.4
##  [13] stringr_1.5.1          pkgconfig_2.0.3        crayon_1.5.2
##  [16] fastmap_1.1.1          labeling_0.4.3         utf8_1.2.4
##  [19] promises_1.3.0         rmarkdown_2.26         ps_1.7.6
##  [22] itertools_0.1-3        tinytex_0.50           purrr_1.0.2
##  [25] xfun_0.43              Rfast_2.1.0            randomForest_4.7-1.1
##  [28] jsonlite_1.8.8         limSolve_1.5.7.1       later_1.3.2
##  [31] parallel_4.4.0         cluster_2.1.6          R6_2.5.1
##  [34] stringi_1.8.3          RColorBrewer_1.1-3     parallelly_1.37.1
##  [37] cellranger_1.1.0       Rcpp_1.0.12            bookdown_0.39
##  [40] iterators_1.0.14       knitr_1.46             snow_0.4-4
##  [43] Matrix_1.7-0           igraph_2.0.3           tidyselect_1.2.1
##  [46] yaml_2.3.8             websocket_1.4.1        codetools_0.2-20
##  [49] processx_3.8.4         listenv_0.9.1          doRNG_1.8.6
##  [52] lattice_0.22-6         tibble_3.2.1           plyr_1.8.9
##  [55] withr_3.0.0            rARPACK_0.11-0         evaluate_0.23
##  [58] signal_1.8-0           speaq_2.7.0            RcppParallel_5.1.7
##  [61] xml2_1.3.6             lpSolve_5.6.20         pillar_1.9.0
##  [64] BiocManager_1.30.22    rngtools_1.5.2         foreach_1.5.2
##  [67] ellipse_0.5.0          pcaPP_2.0-4            generics_0.1.3
##  [70] chromote_0.2.0         munsell_0.5.1          scales_1.3.0
##  [73] globals_0.16.3         glue_1.7.0             tools_4.4.0
##  [76] data.table_1.15.4      SparseM_1.81           RSpectra_0.16-1
##  [79] fs_1.6.4               mvtnorm_1.2-4          cowplot_1.1.3
##  [82] grid_4.4.0             impute_1.77.0          missForest_1.5
##  [85] tidyr_1.3.1            colorspace_2.1-0       cli_3.6.2
##  [88] fansi_1.0.6            mixOmics_6.27.0        corpcor_1.6.10
##  [91] doSNOW_1.0.20          gtable_0.3.5           digest_0.6.35
##  [94] progressr_0.14.0       ggrepel_0.9.5          farver_2.1.1
##  [97] htmltools_0.5.8.1      lifecycle_1.0.4        httr_1.4.7
## [100] MASS_7.3-60.2
```