NMR-based Metabolic Profiling with R

Torben Kimhofer

2018-03-26

This vignette documents a typical pre-processing workflow for NMR-based metabolic profilig using the *MetaboMate* R package. Example data represent Proton NMR spectra of [how many] murine urine samples collected pre and post bariatric surgery. One dimensional spectra were acquired on a 600 MHz Bruker Avance III spectrometer, equipped with a 5 mm triple resonance (TXI) probe operating at 300 K. Further information on sample collection and processing as well as data acquisition can be obtained from the original publication by Jia V. Li *et al.*[1]

Prerequisites

Although not essential for this tutorial, I recommend to install RStudio. RStudio is an open-source integrated development environment for R, which includes a code editor that highlights syntax, enables quick and easy access to help pages, improves workspace management and offers various tools for plotting, history and debugging. Simply put, it makes working with R a lot more efficient.

Data Import and Processing

Reading 1D NMR spectra

The first step is to load NMR spectra into to R workspace. This can be accomplished with the **readBruker()** function, which imput argument is the path to the NMR experiment folder. It them imports all spectra into matrix format. A matrix is a data table where each row represents an observation (here this is a urine NMR spectrum) and each column represents a metabolic feature (ppm variable).

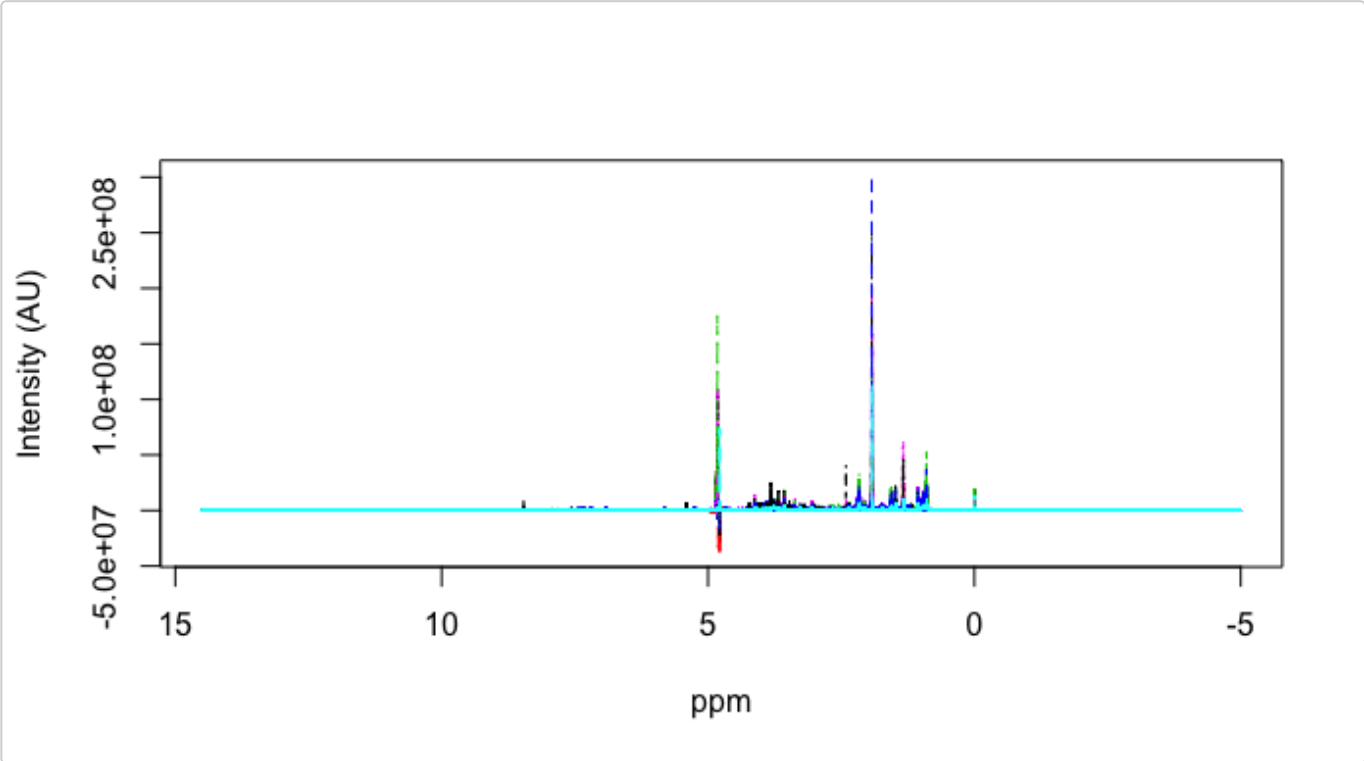
```
library(MetaboMate)
#path=system.file("extdata", package = "MetaboMate")
path='../inst/extdata/'
print(path)
#> [1] "../inst/extdata/"
readBruker(path)
#> Reading 35 spectra.
ls()
#> [1] "meta" "path" "ppm"  "X"
```

The function **readBruker()** searches for all experiment files in a given path and automatically adds the NMR data matrix, ppm and spectrometer metadata (as respective variables X, ppm and meta) to the R workspace. The dataframe **meta** contains all parameters specified in the Bruker *acqus* and *procs* files (rows=specta, columns=parameters).

Plotting spectra

Let's have a first look at all spectra that were imported with the **matspec()** function.

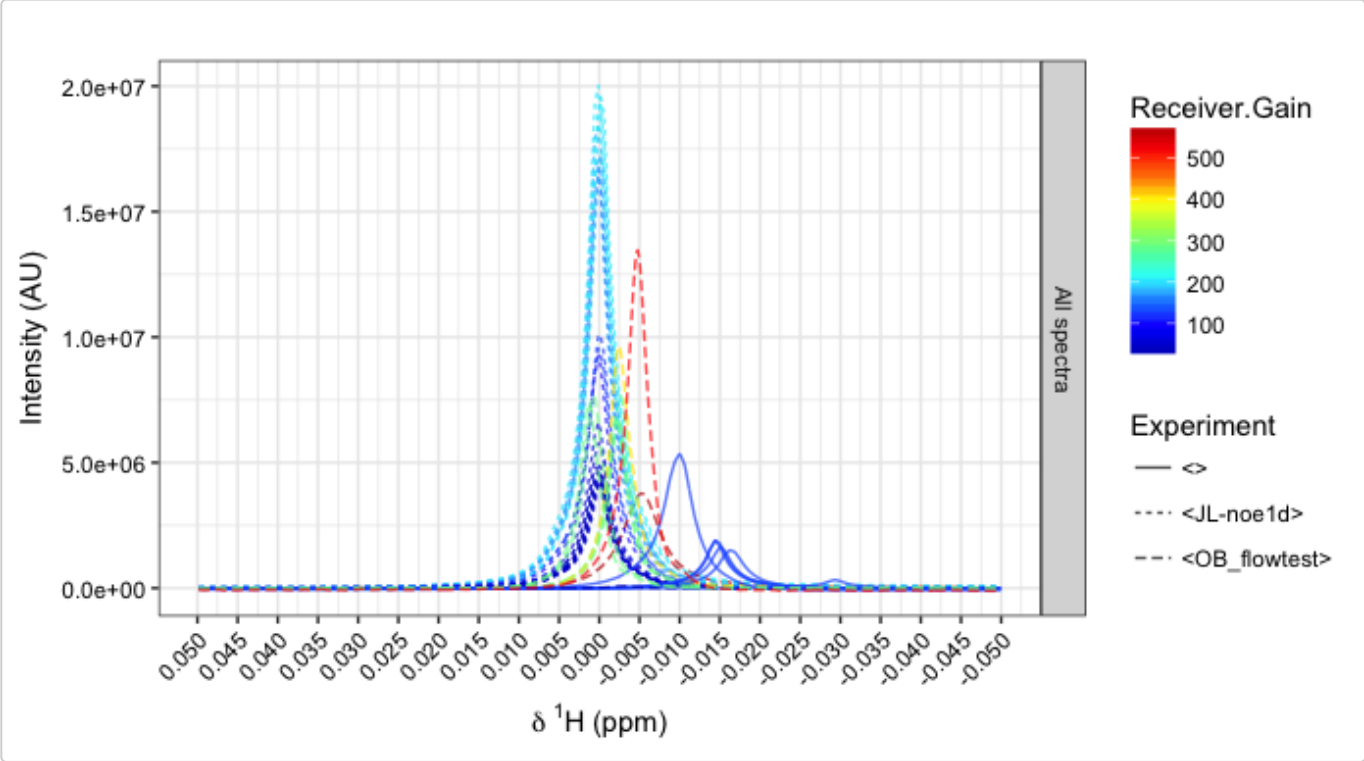
```
matspec(ppm, X, shift = range(ppm))
?matspec
```



The following example illustrates how the experiment metadata can be used. Here, the TSP signal, which is a component of the sample preparation buffer (ie. TSP concentration is equalin all samples) is coloured according to the receiver gain (a parameter that amplifies the NMR signal before it is converted to a digital signal):

```
# plot TSP signal
specOverlay(X, ppm, shift=c(-0.05,0.05),
            an=list('All_Spectra'='All spectra', # facetting (here: a single panel)
                    'Receiver Gain'=as.numeric(meta$a_RG), # colour
                    'Experiment'=factor(meta$a_EXP))) # linetype

# filter X and meta for relevant experiments
table(meta$a_EXP)
#>
#>            <>    <JL-noe1d> <OB_flowtest>
#>             6            20             9
idx <- grep('noe1d', meta$a_EXP)
X <- X[idx,]
meta <- meta[idx,]
```



From the plot above you can see that there were different experiments performed (idicated by different linetypes). We are only interested in the experiment. All others are calibration experiments not suitable for comparative analysis. Therefore, these are excluded (the last comments in the code snitppet above).

Phasing and calibration

Phasing script not available but work in progress.

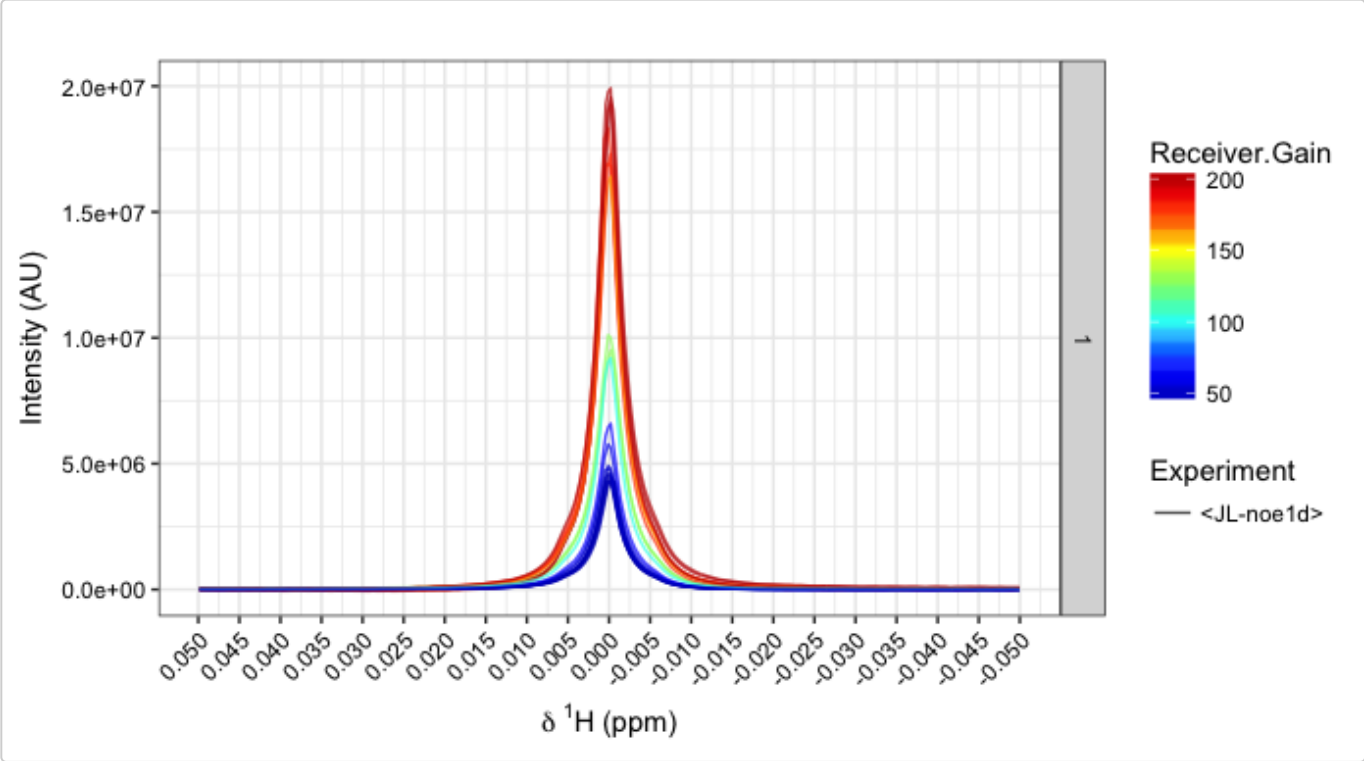
Spectral calibration is an essential step in NMR data processing, where the entire spectrum is shifted until the peak apes of a reference compound is at a certain ppm position. For urine NMR analysis the reference compounds is usually TSP, which is calibrated at zero ppm.[2]

We can calibrate the urine spectra with the following code:

```
# calibrate urinary NMR spectra to TSP
X.cali=calibration(X, ppm, type='Urine')

# plot TSP overlay with calibrated data
specOverlay(X.cali, ppm, shift=c(-0.05,0.05),
            an=list('All_Spectra'=1 ,
                    'Receiver Gain'=as.numeric(meta$a_RG),
                    'Experiment'=factor(meta$a_EXP)))

ls()
#> [1] "idx"    "meta"   "path"   "ppm"    "X"      "X.cali"
```



Now you can see that all TSP signals are nicely aligned with the peak apex centered at zero ppm. By removing non-relevant experiments we can clearly see that the receiver gain (RG) parameter has pronounced effect on specral intensities. In metabolic phenotyping, this parameter is often fixed for spectra acquited within one analytical run as it sometimes can be diffucult to account for.

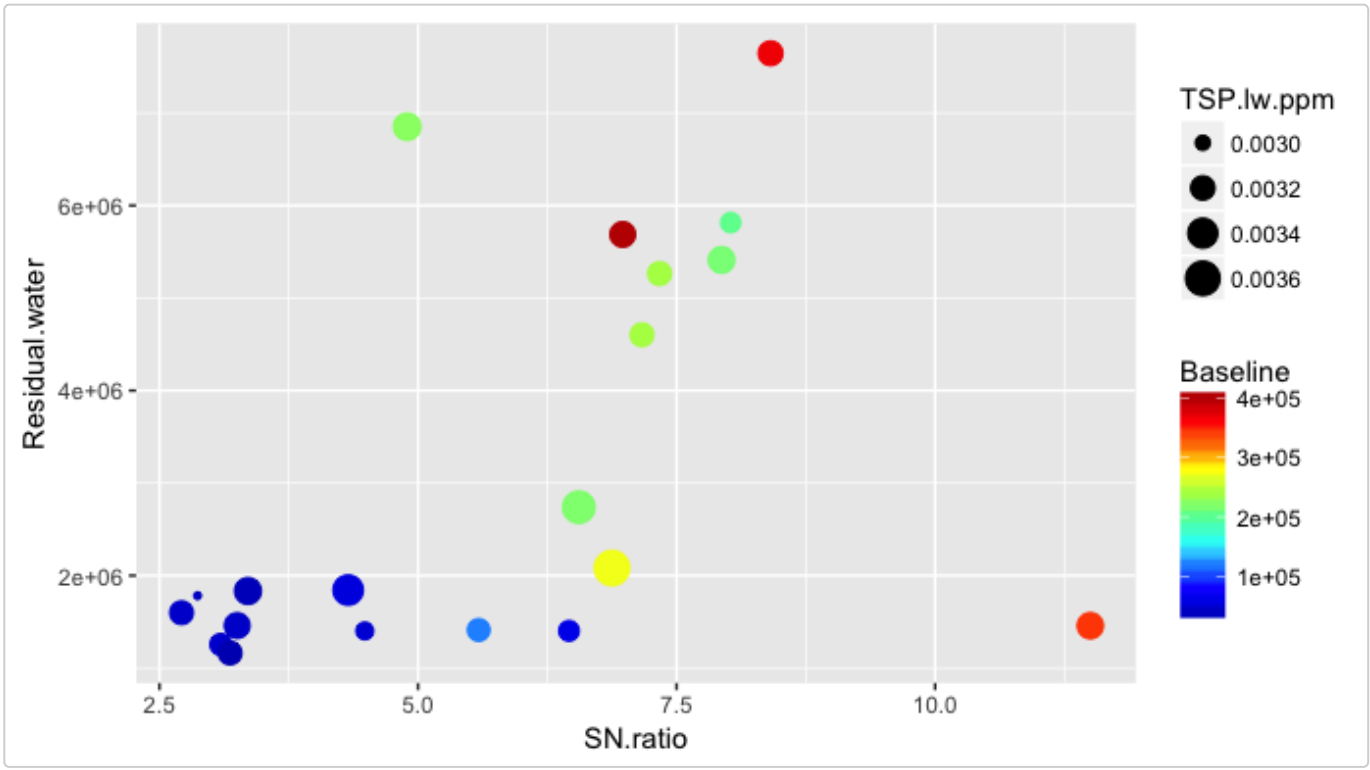
Assessment of spectral quality

In metabolic phenotyping and in any other field where multiple NMR spectra are compared quantitatively, the quality of spectra is of particular importance. Assessment of spectral quality usually is a visual inspection of the water suppresion quality, spectral line widths and baseline stability as well as the average signal to noise (SN) ratio.

High throughput NMR often does not allow a manual inspection of each individual spectrum and automatically generated quality indices are often used to exclude low quality spectra. Here, we derive several spectral quality indices:

```
# calculate quality control measures
spec.qc=spec.quality(X.cali, ppm, ppm.noise=c(9.4,9.5), plot=T)
knitr::kable(head(spec.qc, 6))
```

	TSP.lw.ppm	Residual.water	Baseline	SN.ratio
101	0.00350	2738791	218445.59	6.554764
102	0.00364	2081086	272931.87	6.875540
103	0.00313	1255471	46483.65	3.092912
104	0.00324	5687149	402266.62	6.978859
105	0.00304	1401192	49081.68	4.485429
106	0.00310	1401993	60695.31	6.459934



In the above plot we see that …

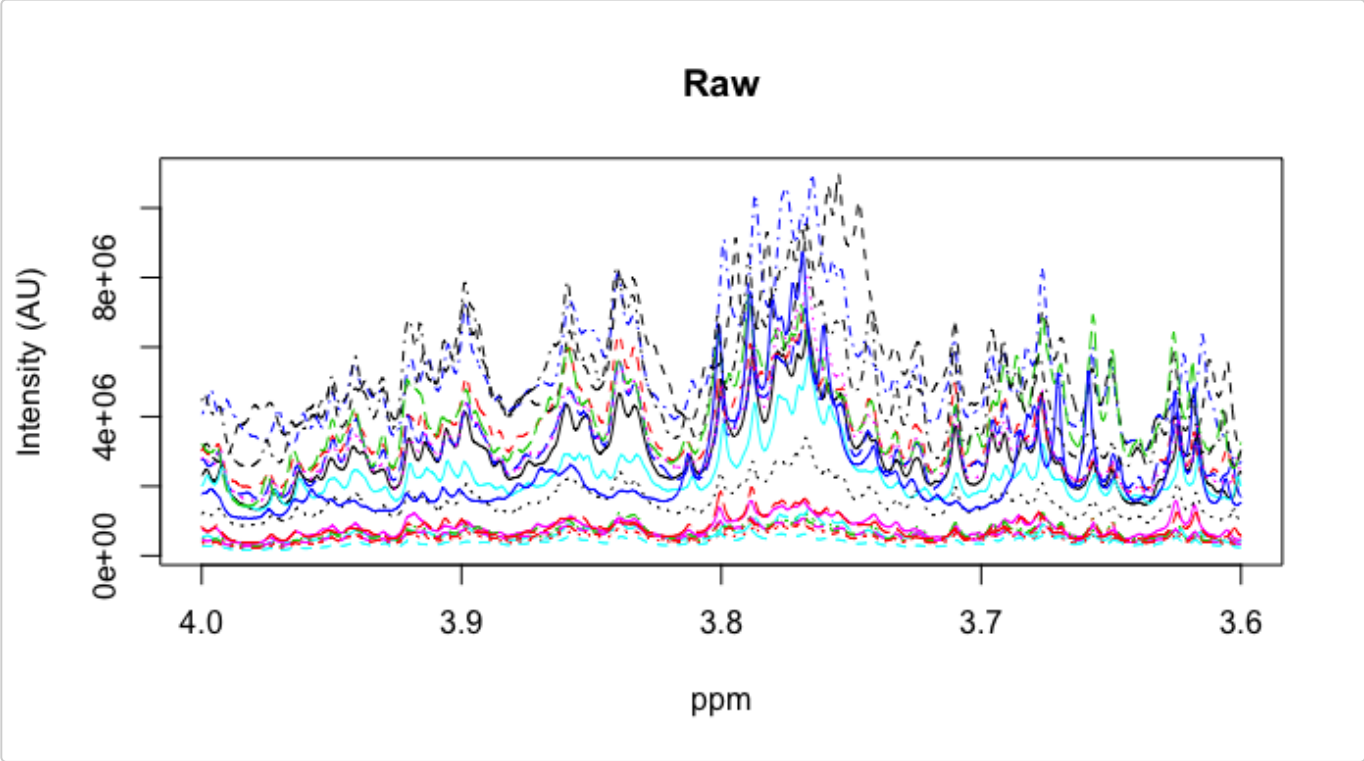
Currently, the quality control line width estimate requires a TSP signal or any other external reference compound that resonates at zero ppm.

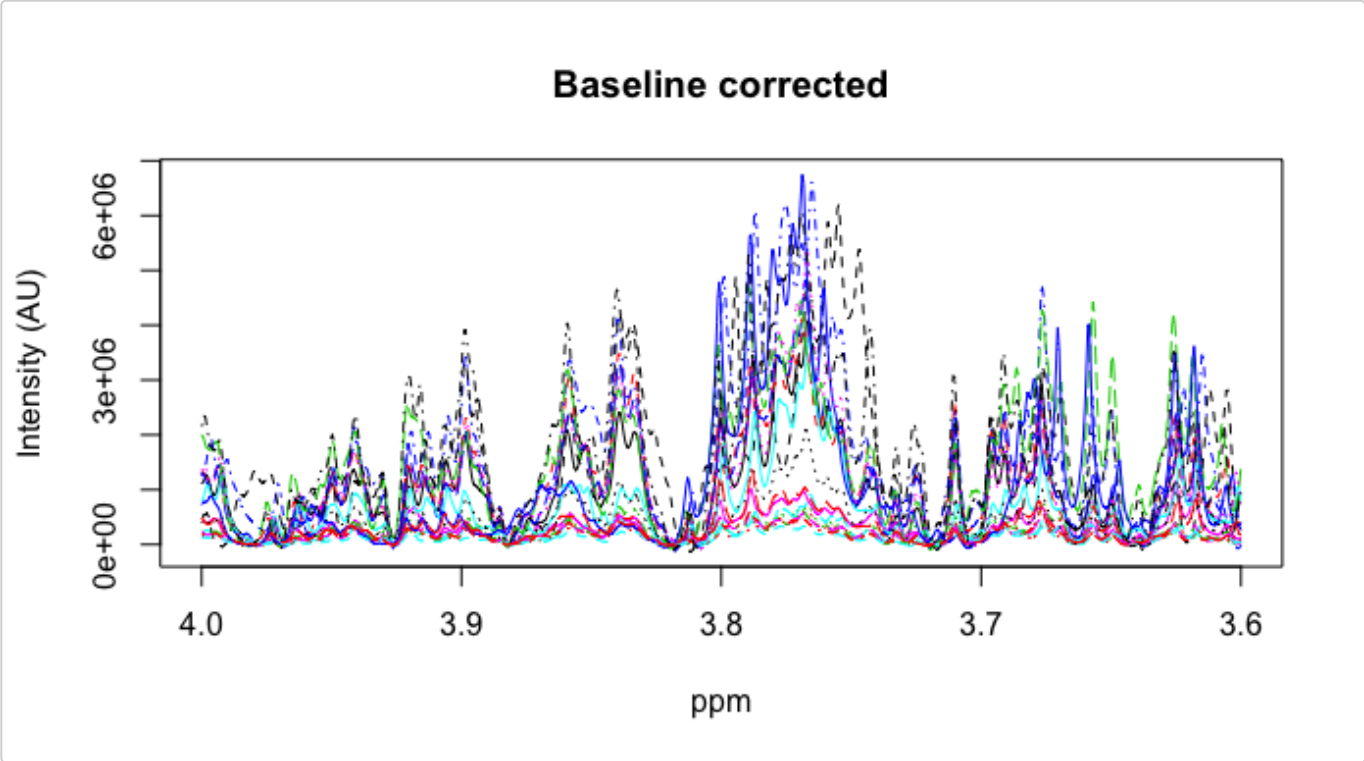
Excision chemical shifts regions and baseline correction

```
# Indexing regions of TSP signal ...
idx.TSP=get.idx(c(min(ppm), 0.5), ppm)
# ...residual water signal
idx.water=get.idx(c(4.6, 5), ppm)
# ... as well as ppm regions without any signals
idx.noiseDF=get.idx(c(9.5, max(ppm)), ppm)
# Excising these regions
X.cali=X.cali[,-c(idx.TSP, idx.water, idx.noiseDF)]
ppm=ppm[-c(idx.TSP, idx.water, idx.noiseDF)]

# Baseline correction
X.bl=bline(X.cali)

# compare spectra before and after baseline correction
matspec(ppm, X.cali, shift=c(3.6,4), main='Raw')
matspec(ppm, X.bl, shift=c(3.6,4), main='Baseline corrected')
```





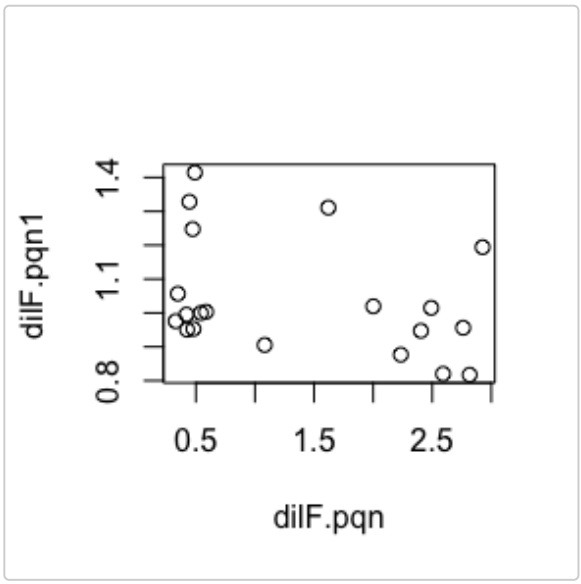
Spectral normalisation

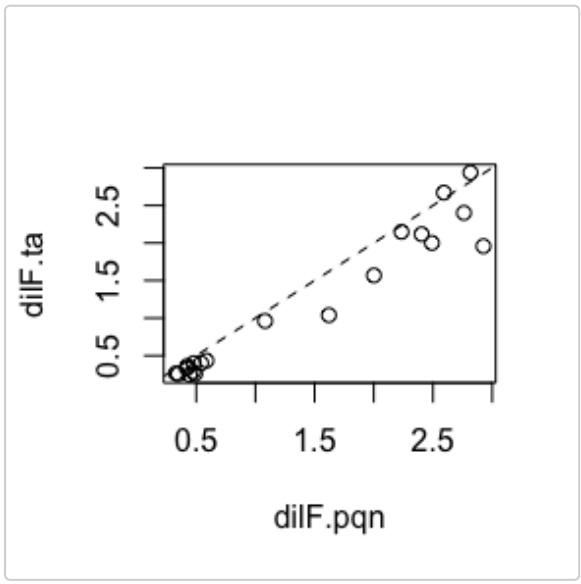
Depending on the sample type, multivariate analysis requires normalisation of spectra. In our urinary NMR study for exmample, the urine sample dilution varies across samples, perhaps due to the uptake of different amounts of water.

There are several different normalisation methods avaiable. Among the most commonly applied are total area normalisaton and probablistic quotient normalisation.[3] In this tutorial, we normalise the samples using both methods for comparative purposes.

```
X.pqn=pqn(X.bl, add.DilF = 'dilF.pqn', TArea = F)
X.pqn.ta=pqn(X.bl, add.DilF = 'dilF.pqn1', TArea = T)
X.ta=totalArea(X.bl, add.DilF='dilF.ta')
```

```
plot(dilF.pqn, dilF.pqn1)
# plot(dilF.pqn1, dilF.ta)
plot(dilF.pqn, dilF.ta)
abline(a = c(0,1), lty=2)
```





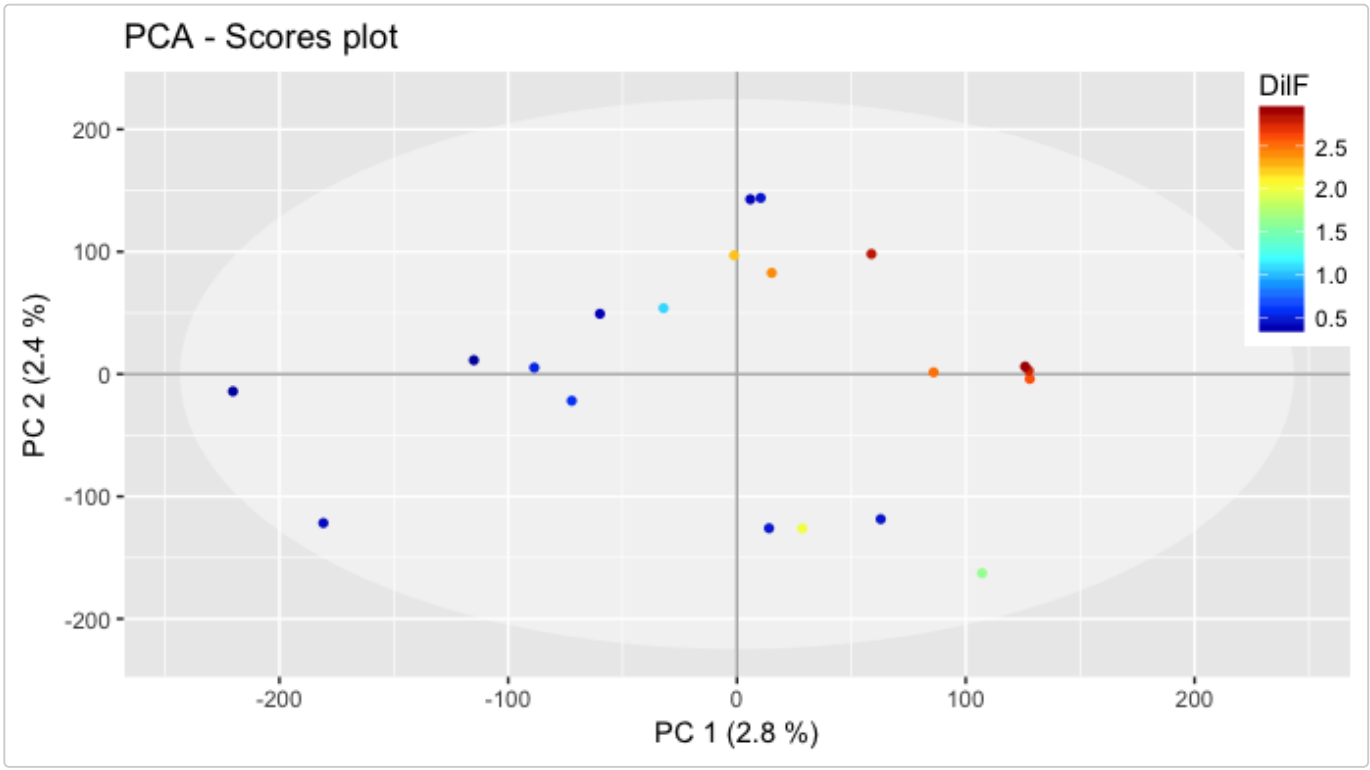
In the present case PQN generates on average slightly lower dilution factors than total area normalisation (the dotted line indicates perfect correlation) with increasing differences towards higher dilutions. Normalisation scales an entire spectrum up or down (multiplication with the dilution factor), and improper dilution factor normalisation can have detrimental effects for further downstream analyses.

Unsupervised Multivariate Analysis

Expoloratory data analysis is usually the first statistical analysis step in metabolic phenotyping. Traditionally, Pricipal Component Analysis (PCA) is used for this step as this projection method summarises the main sources of systematic variation (or latent factors, latent in the sense of unobserved) in so called principal components. Thefore, PCA reduces data dimensionality and can give important insights into the data structure.

To perform PCA and plot the resuls with please execute the code below.

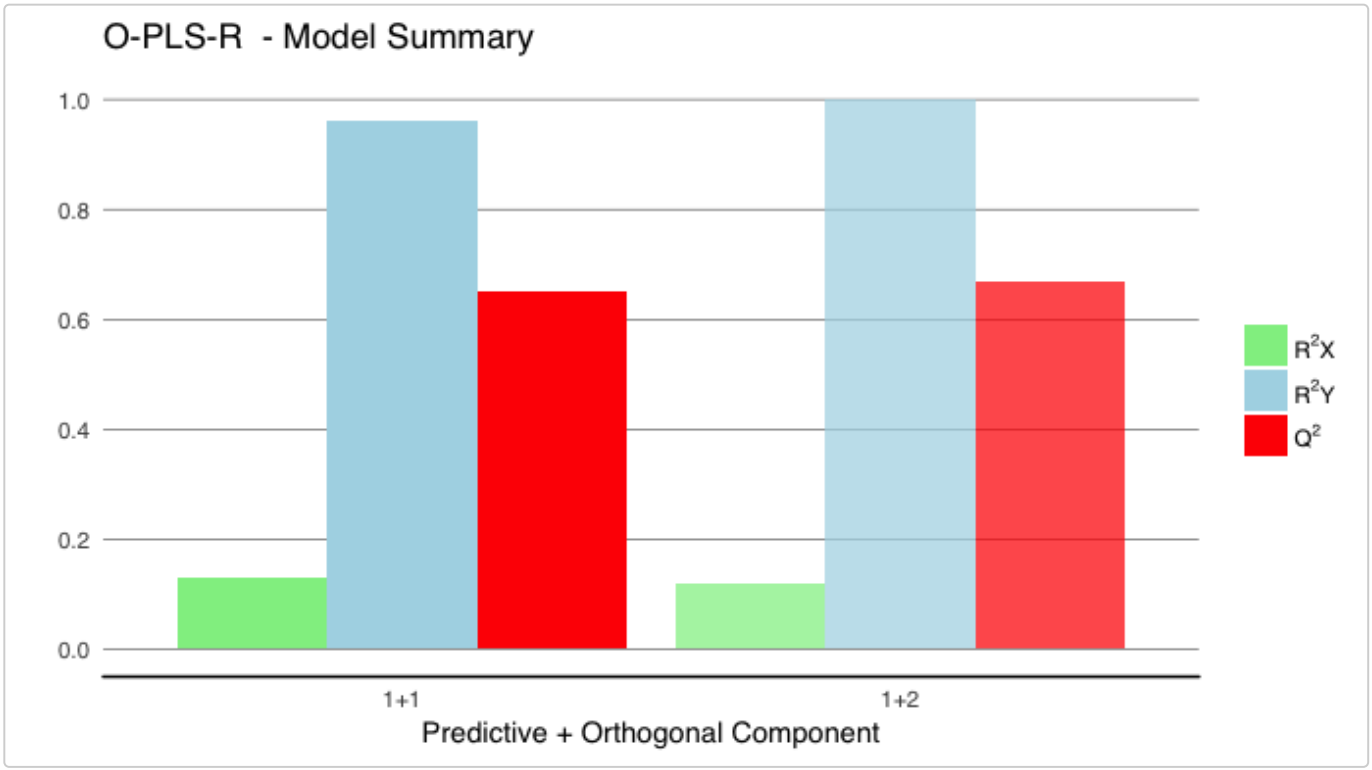
```
pca.model=pca(X=X.pqn, pc=2, scale='UV', center=T)
plotscores(pca.model, pc=c(1,2), an=list(DilF=dilF.pqn), title='PCA - Scores plot')
```

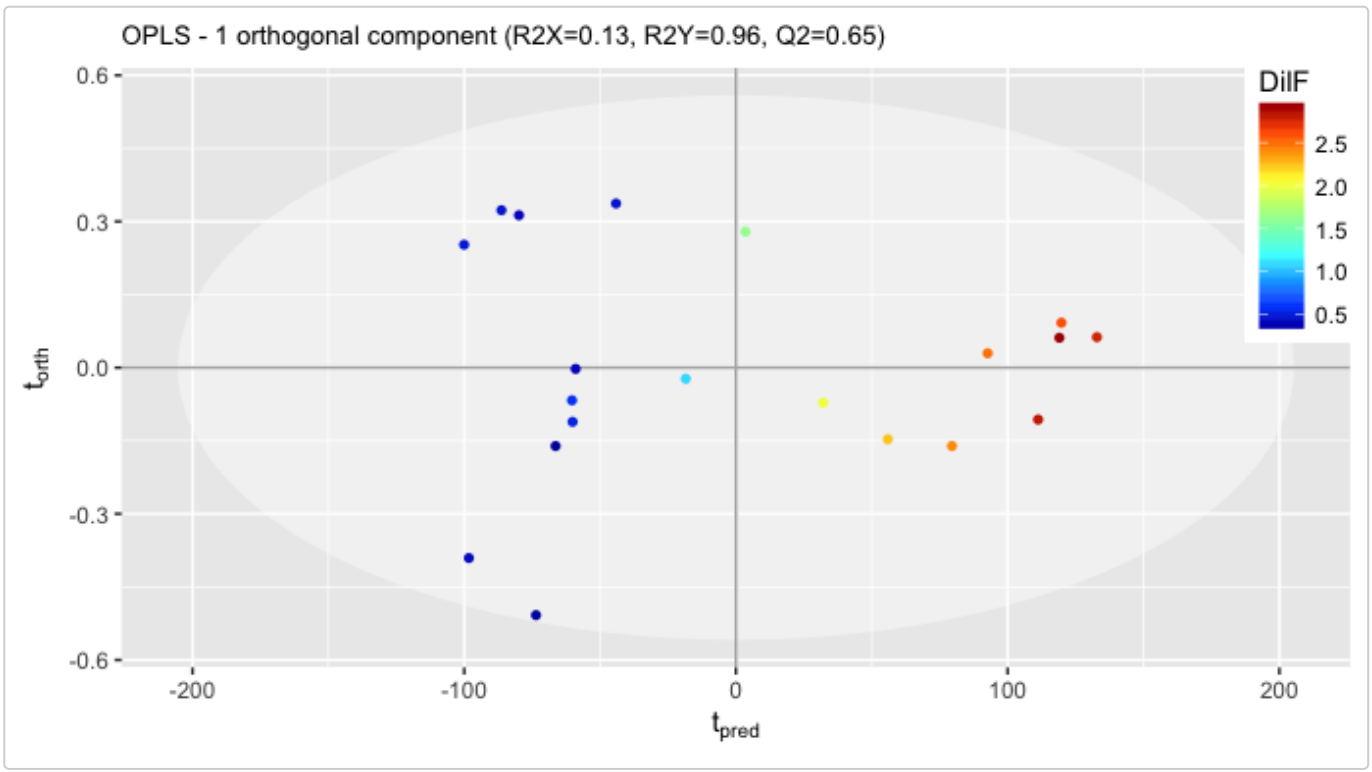


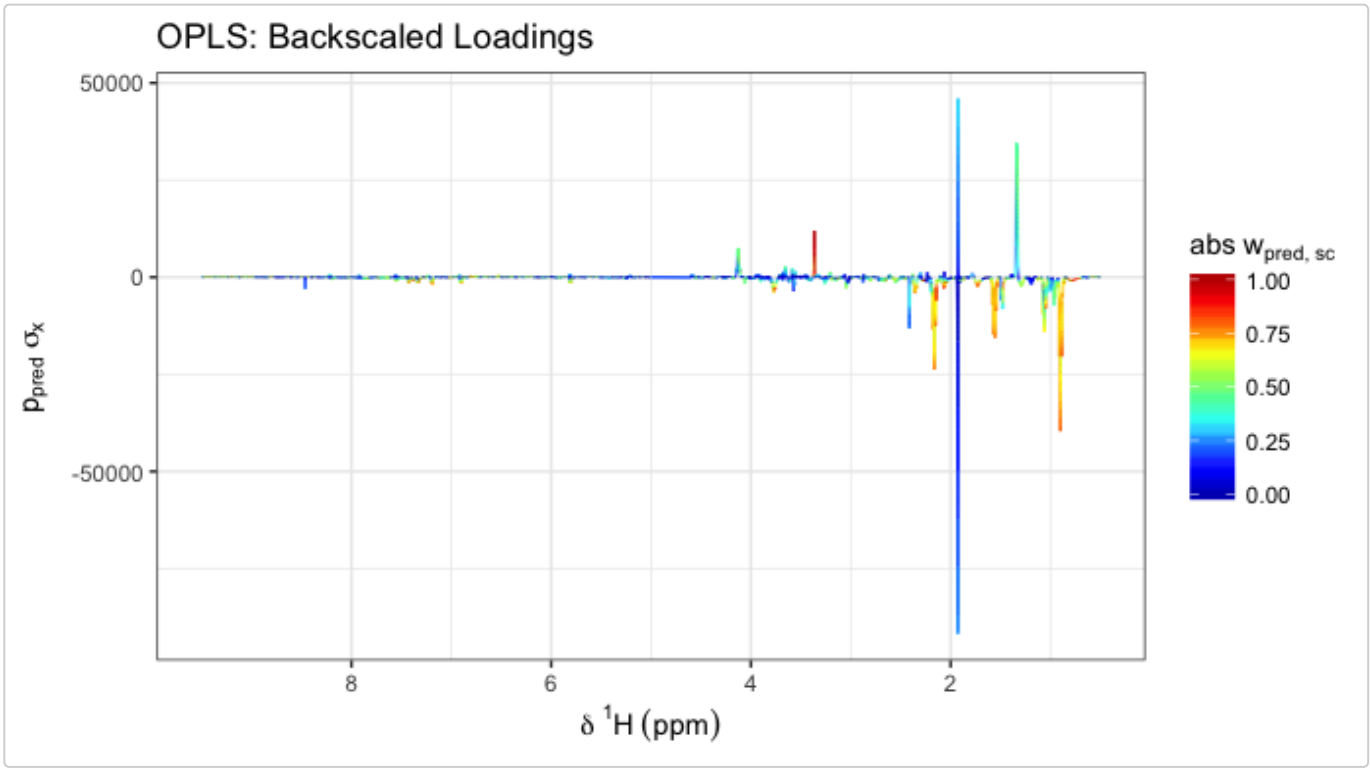
Supervised Multivariate Analysis

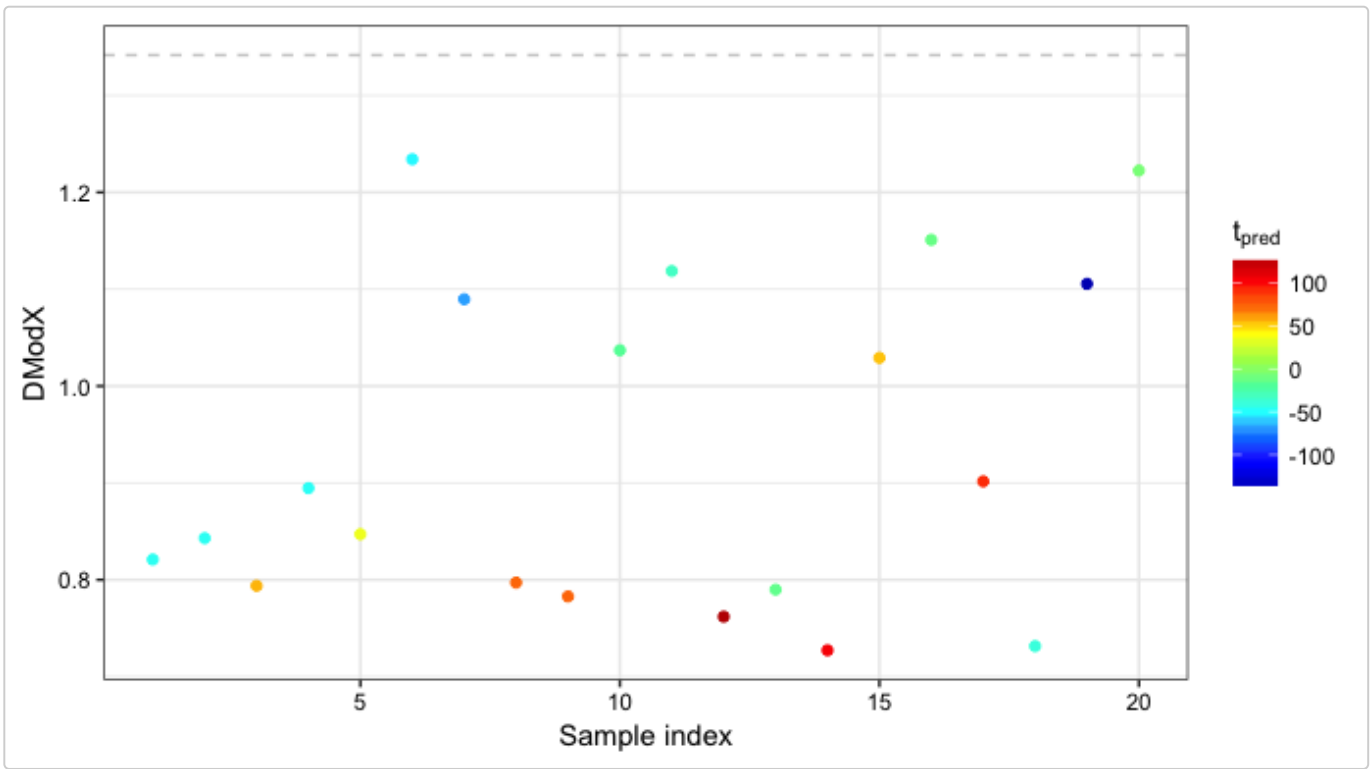
OPLS-DA or PCA-R

```
opls.model=opls(X=X.pqn,Y=dilF.pqn)
#> Preparing data ...
#> Calculate total sum of squares...done
#> Performing OPLS-R ...
#> Fiting components...Component  1 ...Component  2 ...At PC 2: delta Q2 < 0.03: 0.02
#> done
plotscores(opls.model, pc=c(1,2), an=list(DilF=dilF.pqn), title='OPLS - Scores plot', cv.scores = F)
plotload(opls.model, X.pqn, ppm, type='Backscaled', title = 'OPLS: Backscaled Loadings')
distanceX=dmodx(opls.model, plot=T)
```


O-PLS-R - Model Summary
1.0
0.8
0.6
0.4
0.2
0.0
1+1
1+2
Predictive + Orthogonal Component
R2X
R2Y
Q2


OPLS - 1 orthogonal component (R2X=0.13, R2Y=0.96, Q2=0.65)
DilF
2.5
2.0
1.5
1.0
0.5
0.6
0.3
0.0
-0.3
-0.6
torth
-200
-100
0
100
200
tpred

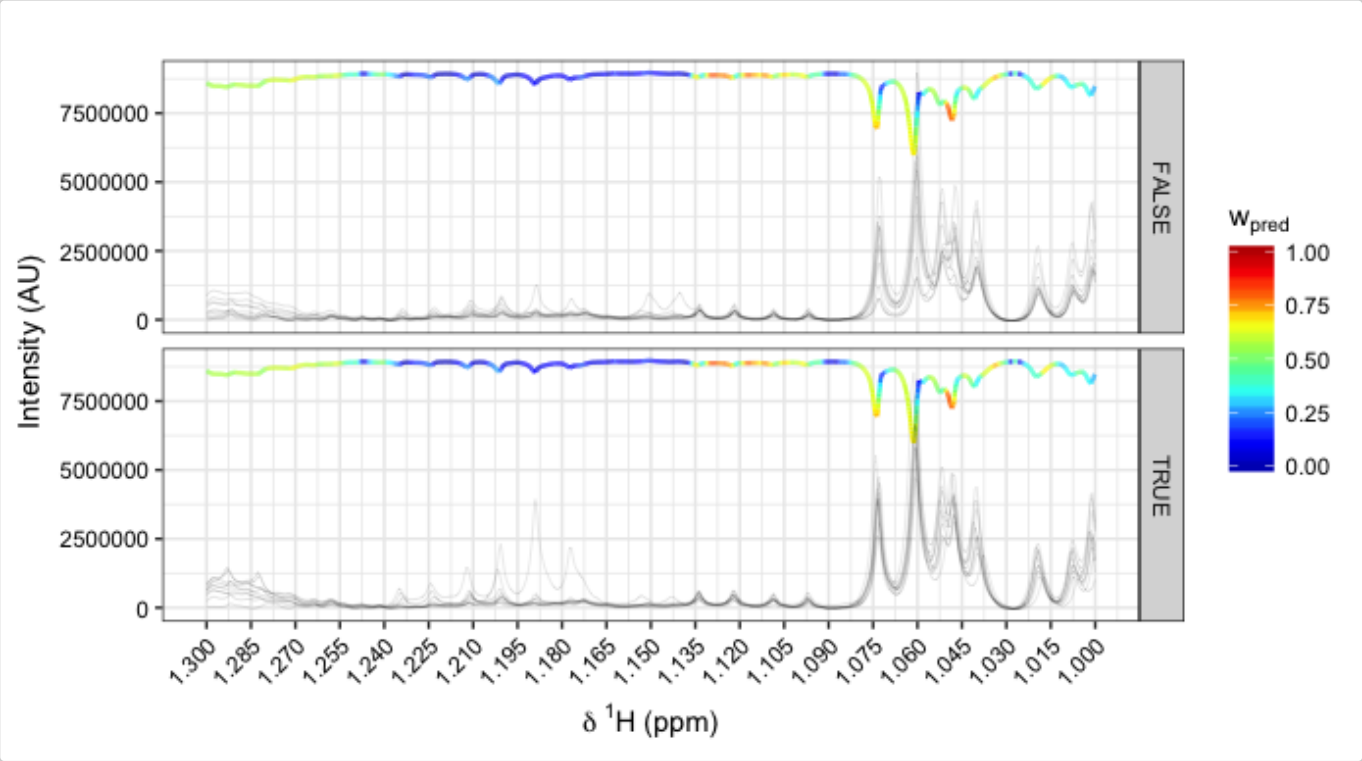




Further plotting options

Metabolite Identification

```
specload(opls.model, X.pqn, ppm, shift=c(1,1.3), an=list(dilF.pqn<1 ), type='Backscaled')
```



Vignette Info

This vignette is not completed. Missing sections: Metabolite Identification, Clustering, several explanations (although man pages should be all available).

---

1. Li, Jia V., *et al.* (2011) Metaboilc surgery profoundly influences gut microbial-host mtaboilc cross-talk. *Gut* 60.9, 1214-1223.↩
2. Dona, A.C., *et al.* (2014) Precision high-throughput proton NMR spectroscopy of human urine, serum, and plasma for large-scale metabolic phenotyping. *Analytical Chemistry*. 86.19. 9887-94.↩
3. Dieterly, F., (2006), Probabilistic Quotient Normalization as Robust Method to Account for Dilution of Complex Biological Mixtures. Application in 1H NMR Metabonomics, , 78.3, 4281-90↩