Code

Specmine - R package for metabolomics and spectral data analysis and mining

Sara Cardoso

2018-04-23

1 Introduction

This package provides a set of methods for metabolomics data analysis, including data loading in different formats, pre-processing, metabolite identification, univariate and multivariate data analysis, machine learning, feature selection and pathway analysis.

A web version, WebSpecmine, is now available, in case you do not have much skills in the R environment or programming, with a easy-to-use interface, with the features implemented in this package and a public repository with metabolomics data (you can also save your own data, private or not, if you create an account). The *specmine* R package may be better to use, if wanted a more flexible use of the provided features and use it with other R tools.

2 Package instalation

So that the package can be fully used and installed correctly, some R packages have to be installed before installing *specmine*

2.1 Installing necessary CRAN packages

2.1.1 Necessary packages that may not be installed

Hide

```
install.packages(c('lattice', 'ggplot2', 'caret', 'BradleyTerry2', 'e1071', 'earth',
                   'fastICA','gam', 'ipred', 'kernlab', 'klaR', 'MASS', 'ellipse', 'mda',
                   'mgcv', 'mlbench','MLmetrics', 'nnet', 'party', 'pls', 'pROC', 'proxy',
                   'randomForest', 'RANN','spls', 'subselect', 'pamr', 'superpc', 'Cubist',
                   'testthat', 'igraph','Rweka', 'stats', 'scatterplot3d', 'compare',
                   'hyperSpec', 'ChemoSpec', 'baseline','rgl', 'Metrics', 'GGally',
                   'ggdendro', 'pcaPP', 'RColorBrewer', 'grid', 'methods', 'qdap',
                   'shinydashboard', 'shinyBS', 'shinyjs', 'DT', 'RMySQL', 'bcrypt',
                   'openssl', 'GGally','shinyWidgets', 'colourpicker', 'devtools',
                   'MLmetrics', 'speaq'))
```

2.2 Installing necessary BioConductor packages

1: If bioconductor is not yet installed:

Hide

```
source("https://bioconductor.org/biocLite.R")
biocLite()
```

2: Bioconductor packages necessary:

Hide

```
source("https://bioconductor.org/biocLite.R")
biocLite("impute", "genefilter", "xcms", "MAIT", "KEGGREST", "KEGGgraph", "mzR")
```

2.3 Installing other packages

Hide

```
devtools::install_github('cytoscape/r-cytoscape.js')
```

2.4 Installing *specmine*

2.4.1 CRAN version

Hide

```
install.packages("specmine")
```

2.4.2 Development version

Note that, at some point, the development version may not be conpletely error free.

Hide

```
devtools::install_bitbucket('chrisbcl/metabolomicspackage', ref='master')
```

3 Supported Data

Various types of data are supported, in many formats. The website considers that **each data file represents one distinct sample, with exception for when one csv file of UV-VIS, IR and Raman Spectra is given and for the data file of concentrations data**.

3.1 NMR and GC/LC-MS Peak Lists

The peak lists data files must have the **CSV format**. Each CSV file must represent a sample and have two columns: the first one corresponds to the chemical shifts (in ppms) or the mass/charge ratios and the second one the intensities of those peaks.

Part of a CSV file **example** of a peak list:

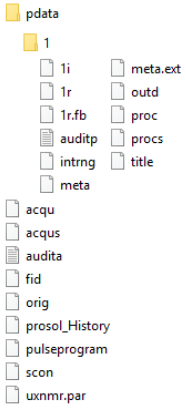
```
ppm,intensity
0.74,0.0001
0.89,0.0004
0.90,0.0007
0.91,0.0005
0.91,0.0008
0.92,0.0004
0.94,0.0003
0.95,0.0004
0.96,0.0009
```

3.2 NMR Spectra

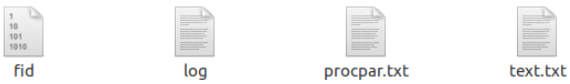
There are two nmr spectra formats that are supported.

The **BRUKER** format is supported, if the *processed spectra* are given. Each spectrum data has to be in a different folder. **Each folder has to have the following structure:**

At least the files procs and 1r have to be present. They have to be inside spectrumfoldername/pdata/1



The **VARIAN** format is supported, only if the raw *fid* file is given, alongside with the *procpar* file. Each spectrum data has to be in a different folder. **Each folder has to have the following structure:**



3.3 GC/LC-MS Spectra

The MS spectral data files must either have **.mzXML**, **.netCDF** or **mzData** formats.

3.4 UV-Vis, IR and Raman Spectra

The data files of these type of spectra must be in one of the following formats: CSV, (J)DX, SPC or MS EXCEL (.xlsx).

For data in **MS EXCEL or CSV files**, each file must have two columns: the first one representing the wavenumber, wavelength or raman shift, according to the type of spectra, and the second one the value of the measurements.

When only **one CSV file** is given, the structure as to be similar to the following example (the first column corresponds to the wavenumber, wavelength or raman shift, according to the type of spectra):

```
,sampleName1,sampleName2
200,0.085956648,0.04830468
201,0.067182627,0.017316359
202,0.044842223,0.026930633
203,0.051335963,0.041539431
```

3.5 Concentrations Data

Concentrations data must be a **CSV or TSV file** with the samples names in the first column (each line then corresponds to a sample) and the concentrations values for each metabolite in the following columns. Alternatively, samples names can be in the first line (each column then corresponds to a sample) and the concentrations values for each metabolite in the following lines.

Part of a CSV example file of concentrations file:

```
Patient ID,1.6-Anhydro-beta-D-glucose,1-Methylnicotinamide,2-Aminobutyrate
PIF_178,40.85,65.37,18.73
PIF_087,62.18,340.36,24.29
PIF_090,270.43,64.72,12.18
NETL_005_V1,154.47,52.98,172.43
PIF_115,22.2,73.7,15.64
```

3.6 Metadata File

As regards to the metadata file, it can either have CSV or TSV format. Each line should correspond to a sample, where the first column represents the names of such samples, and the remaining ones the metadata classes.

The first column corresponds to the names of the samples. **For the cases where more than one data file is given, the names of the samples have to correspond to the names of the data files.**

Here you have an **example** of a metadata file:

```
Sample Name,Seasons
July2010,Winter
September2010,Spring
October2010,Spring
November2010,Spring
February2011,Sum/Aut
March2011,Sum/Aut
April2011,Sum/Aut
may2011,Sum/Aut
June2011,Winter
July2011,Winter
August2011,Winter
September2011,Spring
October2011,Spring
```

4 Read data into specmine

The data files used in the examples decribed in this section can be obtained in here.

4.1 Structure of a specmine dataset

A specmine dataset is a list with the following elements:

- *data*: data frame of the data points. Each column corresponds to a samples and each line to a data variable. The values in each cell are the yy values of said variable in said sample;
- *metadata*: data frame of the metadata information. Each column corresponds to a metadata variable and each line to a samples. The values in each cell are the values of said variable in said sample;
- *type*: string indicating the type of data. It can either be “nmr-peaks”, “nmr-spectra”, “lcms-peaks”, “gcms-peaks”, “lcms-spectra”, “gcms-spectra”, “ir-spectra”, “uvv-spectra”, “raman-spectra”, “integrated-data”, “concentrations”, “undefined”;
- *description*: a short description of the data;
- *labels*: list with the following elements
  - *x*: xx axis labels;
  - *val*: yy axis labels.

4.2 NMR and GC/LC-MS Peak Lists (including peak alignment)

4.2.1 Functions to use

1: Read NMR and MS Peak lists data into list of data samples

> read_csvs_folder(foldername, header=TRUE, sep=“,”, dec=“.”, ...)

- *foldername*: string containing the path of the data folder;
- *header*: boolean value (TRUE or FALSE) indicating whether data files have a header row with the names of the data variables. Defaults to TRUE;
- *sep*: the separator character of the data values. Defaults to “,”;
- *dec*: character used in the file for decimal points. Defaults to “.”;
- ...: additional parameters for read.csv function from utils package.

2: Read metadata file (optional step but recomended)

> read_metadata(filename, header.col = T, header.row = T, sep = “,”)

- *filename* : string indicating the path of the file with the metadata;
- *header.col*: boolean value (TRUE or FALSE) indicating if the metadata CSV file contains a header column with the name of the metadata variables. Defaults to TRUE;
- *header.row*: boolean value (TRUE or FALSE) indicating if the metadata CSV file contains a header row with the name of the samples. Defaults to TRUE;
- *sep*: the separator character. Defaults to “,”.

3: Perform Peak Alignment, into specmine dataset

> group_peaks(sample.list, type, method = “own”, metadata = NULL, samp.classes = 1, description = “”, label.x = NULL, label.values = NULL, step = 0.03)

- *sample.list*: list containing the sample’s data. This list can be obtained from the function *read_csvs_folder* above;

- *type*: type of the data. Can either be “nmr-peaks”, “lcms-peaks” or “gcms-peaks”;
- *method*: method of peak alignment. Can either be
    - “own”: Specmine method. Default value;
    - “metaboanalyst”: MetaboAnalyst method, which is for using the peak alignment used in MetaboAnalyst software.
- *metadata*: data frame containing the metadata. Can be obtained from the function *read_metadata* above; **optional but recomended**
- *samp.classes*: string containg the metadata’s variable name to be used in the MetaboAnalyst method. Can be obtained from colnames(metadata_dataframe). Defaults to the variable represented by the first column.
- *description*: short description of the data. **optional**
- *label.x*: the label for the x values. **optional**
- *label.values*: the label for the y values. **optional**
- *step*: step value for the peak alignment process in the specmine method. Defaults to 0.03.

4.2.2 Example

1. Strings indicating where data and metadata is:

Hide

```
nmr_peaks_lists_data_folder="/home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data"
nmr_peaks_lists_metadata_file="/home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/propolis_nmr_metadata.csv"
```

2. Read data folder:

Hide

```
nmr_peaks_list=read_csvs_folder(nmr_peaks_lists_data_folder)
```

```
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/AC_au.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/AC_sm.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/AC_sp.CSV"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/AC_wi.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/AN_au.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/AN_sm.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/AN_sp.CSV"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/AN_wi.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/BR_au.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/BR_sm.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/BR_sp.CSV"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/BR_wi.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/CE_au.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/CE_sm.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/CE_sp.CSV"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/CE_wi.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/CN_au.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/CN_sm.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/CN_sp.CSV"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/CN_wi.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/DC_au.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/DC_sm.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/DC_sp.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/DC_wi.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/FP_sm.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/FP_sp.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/FP_wi.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/IT_au.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/IT_sm.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/IT_sp.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/JB_au.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/JB_sm.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/JB_sp.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/JB_wi.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/PU_au.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/PU_sm.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/PU_sp.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/PU_wi.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/SA_au.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/SA_sm.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/SA_sp.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/SA_wi.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/SJ_au.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/SJC_au.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/SJC_sm.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/SJC_sp.CSV"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/SJ_sm.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/SJ_sp.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/SJ_wi.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/UR_au.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/UR_sm.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/UR_sp.CSV"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/UR_wi.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/VR_au.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/VR_sm.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/VR_sp.CSV"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/XX_au.csv"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/XX_sm.CSV"
## [1] "Reading sample  /home/scardoso/Documents/metabolomics_datasets/NMR/Peak lists/propolis/Propolis_NMR Data_data/XX_wi.csv"
```

3. Read metadata file:

Hide

```
nmr_peaks_metadata=read_metadata(nmr_peaks_lists_metadata_file)
```

4. Align peaks using specmine method with a step of 0.03 (default values, so it is not necessary to define them), which will now return the specmine dataset:

Hide

```
nmr_peaks_dataset=group_peaks(nmr_peaks_list, "nmr-peaks", metadata = nmr_peaks_metadata, description="propolis nmr samples", label.x
= "ppm", label.values = "intensity")
```

5. Table of the data just loaded (it is possible to realize that there are missing values):

Hide

```
DT::datatable(nmr_peaks_dataset$data, options=list(scrollX = TRUE))
```

6. Table of the metadata just loaded:

Hide

```
DT::datatable(nmr_peaks_dataset$metadata)
```

4.3 NMR Spectra (peak detection independent of data reading)

4.3.1 Functions to use

4.3.1.1 When the data is in BRUKER format

> read_Bruker_files(bruker_directory, metadata_file=NULL, m.header_col=T, m.header_row=T, m.sep=",", samples.names=NULL, zipped=T, description="", label.x="ppm", label.values="intensity")

- *bruker_directory*: string containing the path of the data folder with all the spectra folders;
- *metadata_file*: string containing the path of the metadata file; **optional but recomended**
- *m.header_col*: boolean value (TRUE or FALSE) indicating if the metadata file contains a header column with the name of the metadata variables. Defaults to TRUE;
- *m.header_row*: boolean value (TRUE or FALSE) indicating if the metadata file contains a header row with the name of the samples. Defaults to TRUE;
- *m.sep*: the separator character of the metadata file. Defaults to ",";
- *samples.names*: CSV file where the first column represents the samples names and in the second column the names of the spectra directories to which they correspond. If NULL, it will be considered that the directories names are the samples names (it has to be the same names that appear in the metadata file);
- *zipped*: Boleean value (TRUE or FALSE) indicating if the spectra directories are zipped (.zip) or not. **The compressed files must have the extension .zip. If not, you will have to uncompress them yourself.**
- *description*: a short text describing the dataset. **optional**
- *label.x*: the label for the x values. **optional**
- *label.values*: the label for the y values. **optional**

4.3.1.2 When the data is VARIAN format

TO USE THIS FUNCTION YOU MUST HAVE PYTHON3 INSTALLED, WITH THE MODULE nmrglue INSTALLED

> read_varian_spectra_raw(varian_spectra_directory, metadata_file=NULL, m.header_col=T, m.header_row=T, m.sep=",", samples.names=NULL, zero_filling=T, apodization=T, zipped=T, description="", label.x="ppm", label.values="intensity")

- *varian_spectra_directory*: string containing the path of the data folder with all the spectra folders;
- *metadata_file*: string containing the path of the metadata file; **optional but recomended**
- *m.header_col*: boolean value (TRUE or FALSE) indicating if the metadata file contains a header column with the name of the metadata variables. Defaults to TRUE;
- *m.header_row*: boolean value (TRUE or FALSE) indicating if the metadata file contains a header row with the name of the samples. Defaults to TRUE;
- *m.sep*: the separator character of the metadata file. Defaults to ",";
- *samples.names*: CSV file where the first column represents the samples names and in the second column the names of the spectra directories to which they correspond. If NULL, it will be considered that the directories names are the samples names (it has to be the same names that appear in the metadata file);
- *zero_filling*: boolean value (TRUE or FALSE) indicating whether zero-filling should be performed or not when processing the fid spectra. Defaults to TRUE;
- *apodization*: boolean value (TRUE or FALSE) indicating whether apodization should be performed or not when processing the fid spectra. Defaults to TRUE;
- *zipped*: Boleean value (TRUE or FALSE) indicating if the spectra directories are zipped (.zip) or not. **The compressed files must have the extension .zip. If not, you will have to uncompress them yourself.**
- *description*: a short text describing the dataset. **optional**
- *label.x*: the label for the x values. **optional**
- *label.values*: the label for the y values. **optional**

4.3.1.3 After reading spectra, detection and alignment of peaks can be performed, as it is not obligatory when reading the data, which happens when reading MS spectra

Detection of peaks, followed by alignment of those peaks

> detect_nmr_peaks_from_dataset(dataset, baseline_tresh=50000, ap.method="own", ap.samp.classes=1, ap.step=0.03)

- *dataset*: a specmine dataset of type *nmr-spectra*;
- *baseline_tresh*: Minimum intensity value that peaks must have. Peaks with intensity smaller than baseline_tresh will not be considered as detected peaks. Defaults to 50000;
- *ap.method*: method of peak alignment. Can either be
  - "own": Specmine method. Default value;
  - "metaboanalyst": MetaboAnalyst method, which is for using the peak alignment used in MetaboAnalyst software.

- *ap.samp.classes*: string containg the metadata's variable name to be used in the MetaboAnalyst method. Can be obtained from colnames(metadata_dataframe). Defaults to the variable represented by the first column.
- *ap.step*: step value for the peak alignment process in the specmine method. Defaults to 0.03.

4.3.2 Examples

4.3.2.1 BRUKER data

The example data here used was obtained from the Metabolights database, under the ID MTBLS151. The data was obtained using the *get_metabolights_study* function.

1. Strings indicating where data and metadata is:

Hide

```
bruker_nmr_spectra_folder="/home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS151"
bruker_nmr_metadata_file="/home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS151/metadata.csv"
```

2. Loading NMR bruker spectral data to specmine:

In this data, the spectra folders' names correspond to the names of the samples, so no file will be given to the argument samples.names. These folders are zipped (.zip)

Hide

```
nmr_bruker_spectra_dataset=read_Bruker_files(bruker_nmr_spectra_folder, metadata_file=bruker_nmr_metadata_file)
```

```
## Reading Metadata file
## Reading sample  L-IL308C1-1  in  /home/scardoso/temp/L-IL308C1-1/L-IL308C1-1/10/pdata/1
## Reading sample  L-IL308C1-2  in  /home/scardoso/temp/L-IL308C1-2/L-IL308C1-2/10/pdata/1
## Reading sample  L-IL308C1-3  in  /home/scardoso/temp/L-IL308C1-3/L-IL308C1-3/10/pdata/1
## Reading sample  L-IL308C2-1  in  /home/scardoso/temp/L-IL308C2-1/L-IL308C2-1/10/pdata/1
## Reading sample  L-IL308C2-2  in  /home/scardoso/temp/L-IL308C2-2/L-IL308C2-2/10/pdata/1
## Reading sample  L-IL308C2-3  in  /home/scardoso/temp/L-IL308C2-3/L-IL308C2-3/10/pdata/1
## Reading sample  L-IL308C3-1  in  /home/scardoso/temp/L-IL308C3-1/L-IL308C3-1/10/pdata/1
## Reading sample  L-IL308C3-2  in  /home/scardoso/temp/L-IL308C3-2/L-IL308C3-2/10/pdata/1
## Reading sample  L-IL308C3-3  in  /home/scardoso/temp/L-IL308C3-3/L-IL308C3-3/10/pdata/1
## Reading sample  L-IL308C4-1  in  /home/scardoso/temp/L-IL308C4-1/L-IL308C4-1/10/pdata/1
## Reading sample  L-IL308C4-2  in  /home/scardoso/temp/L-IL308C4-2/L-IL308C4-2/10/pdata/1
## Reading sample  L-IL308C4-3  in  /home/scardoso/temp/L-IL308C4-3/L-IL308C4-3/10/pdata/1
## Reading sample  L-IL308C8-1  in  /home/scardoso/temp/L-IL308C8-1/L-IL308C8-1/10/pdata/1
## Reading sample  L-IL308C8-2  in  /home/scardoso/temp/L-IL308C8-2/L-IL308C8-2/10/pdata/1
## Reading sample  L-IL308C8-3  in  /home/scardoso/temp/L-IL308C8-3/L-IL308C8-3/10/pdata/1
## Reading sample  L-IL308T1-1  in  /home/scardoso/temp/L-IL308T1-1/L-IL308T1-1/10/pdata/1
## Reading sample  L-IL308T1-2  in  /home/scardoso/temp/L-IL308T1-2/L-IL308T1-2/10/pdata/1
## Reading sample  L-IL308T1-3  in  /home/scardoso/temp/L-IL308T1-3/L-IL308T1-3/10/pdata/1
## Reading sample  L-IL308T2-1  in  /home/scardoso/temp/L-IL308T2-1/L-IL308T2-1/10/pdata/1
## Reading sample  L-IL308T2-2  in  /home/scardoso/temp/L-IL308T2-2/L-IL308T2-2/10/pdata/1
## Reading sample  L-IL308T2-3  in  /home/scardoso/temp/L-IL308T2-3/L-IL308T2-3/10/pdata/1
## Reading sample  L-IL308T3-1  in  /home/scardoso/temp/L-IL308T3-1/L-IL308T3-1/10/pdata/1
## Reading sample  L-IL308T3-2  in  /home/scardoso/temp/L-IL308T3-2/L-IL308T3-2/10/pdata/1
## Reading sample  L-IL308T3-3  in  /home/scardoso/temp/L-IL308T3-3/L-IL308T3-3/10/pdata/1
## Reading sample  L-IL308T4-1  in  /home/scardoso/temp/L-IL308T4-1/L-IL308T4-1/10/pdata/1
## Reading sample  L-IL308T4-2  in  /home/scardoso/temp/L-IL308T4-2/L-IL308T4-2/10/pdata/1
## Reading sample  L-IL308T4-3  in  /home/scardoso/temp/L-IL308T4-3/L-IL308T4-3/10/pdata/1
## Reading sample  L-IL308T8-1  in  /home/scardoso/temp/L-IL308T8-1/L-IL308T8-1/10/pdata/1
## Reading sample  L-IL308T8-2  in  /home/scardoso/temp/L-IL308T8-2/L-IL308T8-2/10/pdata/1
## Reading sample  L-IL308T8-3  in  /home/scardoso/temp/L-IL308T8-3/L-IL308T8-3/10/pdata/1
## Reading sample  L-IL7C1-1  in  /home/scardoso/temp/L-IL7C1-1/L-IL7C1-1/10/pdata/1
## Reading sample  L-IL7C1-2  in  /home/scardoso/temp/L-IL7C1-2/L-IL7C1-2/10/pdata/1
## Reading sample  L-IL7C1-3  in  /home/scardoso/temp/L-IL7C1-3/L-IL7C1-3/10/pdata/1
## Reading sample  L-IL7C2-1  in  /home/scardoso/temp/L-IL7C2-1/L-IL7C2-1/10/pdata/1
## Reading sample  L-IL7C2-2  in  /home/scardoso/temp/L-IL7C2-2/L-IL7C2-2/10/pdata/1
## Reading sample  L-IL7C2-3  in  /home/scardoso/temp/L-IL7C2-3/L-IL7C2-3/10/pdata/1
## Reading sample  L-IL7C3-1  in  /home/scardoso/temp/L-IL7C3-1/L-IL7C3-1/10/pdata/1
## Reading sample  L-IL7C3-2  in  /home/scardoso/temp/L-IL7C3-2/L-IL7C3-2/10/pdata/1
## Reading sample  L-IL7C3-3  in  /home/scardoso/temp/L-IL7C3-3/L-IL7C3-3/10/pdata/1
## Reading sample  L-IL7C4-1  in  /home/scardoso/temp/L-IL7C4-1/L-IL7C4-1/10/pdata/1
## Reading sample  L-IL7C4-2  in  /home/scardoso/temp/L-IL7C4-2/L-IL7C4-2/10/pdata/1
## Reading sample  L-IL7C4-3  in  /home/scardoso/temp/L-IL7C4-3/L-IL7C4-3/11/pdata/1
## Reading sample  L-IL7C8-1  in  /home/scardoso/temp/L-IL7C8-1/L-IL7C8-1/10/pdata/1
## Reading sample  L-IL7C8-2  in  /home/scardoso/temp/L-IL7C8-2/L-IL7C8-2/10/pdata/1
## Reading sample  L-IL7C8-3  in  /home/scardoso/temp/L-IL7C8-3/L-IL7C8-3/10/pdata/1
## Reading sample  L-IL7T1-1  in  /home/scardoso/temp/L-IL7T1-1/L-IL7T1-1/10/pdata/1
## Reading sample  L-IL7T1-2  in  /home/scardoso/temp/L-IL7T1-2/L-IL7T1-2/10/pdata/1
## Reading sample  L-IL7T1-3  in  /home/scardoso/temp/L-IL7T1-3/L-IL7T1-3/10/pdata/1
## Reading sample  L-IL7T2-1  in  /home/scardoso/temp/L-IL7T2-1/L-IL7T2-1/10/pdata/1
## Reading sample  L-IL7T2-2  in  /home/scardoso/temp/L-IL7T2-2/L-IL7T2-2/10/pdata/1
## Reading sample  L-IL7T2-3  in  /home/scardoso/temp/L-IL7T2-3/L-IL7T2-3/10/pdata/1
## Reading sample  L-IL7T3-1  in  /home/scardoso/temp/L-IL7T3-1/L-IL7T3-1/10/pdata/1
## Reading sample  L-IL7T3-1  in  /home/scardoso/temp/L-IL7T3-1/L-IL7T3-1/20/pdata/1
## Reading sample  L-IL7T3-2  in  /home/scardoso/temp/L-IL7T3-2/L-IL7T3-2/10/pdata/1
## Reading sample  L-IL7T3-3  in  /home/scardoso/temp/L-IL7T3-3/L-IL7T3-3/10/pdata/1
## Reading sample  L-IL7T4-1  in  /home/scardoso/temp/L-IL7T4-1/L-IL7T4-1/10/pdata/1
## Reading sample  L-IL7T4-2  in  /home/scardoso/temp/L-IL7T4-2/L-IL7T4-2/10/pdata/1
## Reading sample  L-IL7T4-3  in  /home/scardoso/temp/L-IL7T4-3/L-IL7T4-3/10/pdata/1
## Reading sample  L-IL7T8-1  in  /home/scardoso/temp/L-IL7T8-1/L-IL7T8-1/10/pdata/1
## Reading sample  L-IL7T8-2  in  /home/scardoso/temp/L-IL7T8-2/L-IL7T8-2/10/pdata/1
## Reading sample  L-IL7T8-3  in  /home/scardoso/temp/L-IL7T8-3/L-IL7T8-3/10/pdata/1
## Reading sample  L-KDC1-1  in  /home/scardoso/temp/L-KDC1-1/L-KDC1-1/10/pdata/1
## Reading sample  L-KDC1-2  in  /home/scardoso/temp/L-KDC1-2/L-KDC1-2/10/pdata/1
## Reading sample  L-KDC1-3  in  /home/scardoso/temp/L-KDC1-3/L-KDC1-3/10/pdata/1
## Reading sample  L-KDC2-1  in  /home/scardoso/temp/L-KDC2-1/L-KDC2-1/10/pdata/1
## Reading sample  L-KDC2-2  in  /home/scardoso/temp/L-KDC2-2/L-KDC2-2/10/pdata/1
## Reading sample  L-KDC2-3  in  /home/scardoso/temp/L-KDC2-3/L-KDC2-3/10/pdata/1
## Reading sample  L-KDC3-1  in  /home/scardoso/temp/L-KDC3-1/L-KDC3-1/10/pdata/1
## Reading sample  L-KDC3-2  in  /home/scardoso/temp/L-KDC3-2/L-KDC3-2/10/pdata/1
```

```
## Reading sample  L-KDC3-3  in  /home/scardoso/temp/L-KDC3-3/L-KDC3-3/10/pdata/1
## Reading sample  L-KDC4-1  in  /home/scardoso/temp/L-KDC4-1/L-KDC4-1/10/pdata/1
## Reading sample  L-KDC4-2  in  /home/scardoso/temp/L-KDC4-2/L-KDC4-2/10/pdata/1
## Bruker file does not exist in datapath, or other problems with bruker files...
## Reading sample  L-KDC4-2  in  /home/scardoso/temp/L-KDC4-2/L-KDC4-2/20/pdata/1
## Reading sample  L-KDC4-3  in  /home/scardoso/temp/L-KDC4-3/L-KDC4-3/10/pdata/1
## Reading sample  L-KDC8-1  in  /home/scardoso/temp/L-KDC8-1/L-KDC8-1/10/pdata/1
## Reading sample  L-KDC8-2  in  /home/scardoso/temp/L-KDC8-2/L-KDC8-2/10/pdata/1
## Reading sample  L-KDC8-3  in  /home/scardoso/temp/L-KDC8-3/L-KDC8-3/10/pdata/1
## Reading sample  L-KDT1-1  in  /home/scardoso/temp/L-KDT1-1/L-KDT1-1/10/pdata/1
## Reading sample  L-KDT1-2  in  /home/scardoso/temp/L-KDT1-2/L-KDT1-2/10/pdata/1
## Reading sample  L-KDT1-3  in  /home/scardoso/temp/L-KDT1-3/L-KDT1-3/10/pdata/1
## Reading sample  L-KDT2-1  in  /home/scardoso/temp/L-KDT2-1/L-KDT2-1/10/pdata/1
## Reading sample  L-KDT2-2  in  /home/scardoso/temp/L-KDT2-2/L-KDT2-2/10/pdata/1
## Reading sample  L-KDT2-3  in  /home/scardoso/temp/L-KDT2-3/L-KDT2-3/10/pdata/1
## Reading sample  L-KDT3-1  in  /home/scardoso/temp/L-KDT3-1/L-KDT3-1/10/pdata/1
## Reading sample  L-KDT3-2  in  /home/scardoso/temp/L-KDT3-2/L-KDT3-2/10/pdata/1
## Reading sample  L-KDT3-3  in  /home/scardoso/temp/L-KDT3-3/L-KDT3-3/10/pdata/1
## Reading sample  L-KDT4-1  in  /home/scardoso/temp/L-KDT4-1/L-KDT4-1/10/pdata/1
## Reading sample  L-KDT4-2  in  /home/scardoso/temp/L-KDT4-2/L-KDT4-2/10/pdata/1
## Reading sample  L-KDT4-3  in  /home/scardoso/temp/L-KDT4-3/L-KDT4-3/10/pdata/1
## Reading sample  L-KDT8-1  in  /home/scardoso/temp/L-KDT8-1/L-KDT8-1/10/pdata/1
## Reading sample  L-KDT8-2  in  /home/scardoso/temp/L-KDT8-2/L-KDT8-2/10/pdata/1
## Reading sample  L-KDT8-3  in  /home/scardoso/temp/L-KDT8-3/L-KDT8-3/10/pdata/1
## Reading sample  R-IL308C1  in  /home/scardoso/temp/R-IL308C1/R-IL308C1/10/pdata/1
## Reading sample  R-IL308C2  in  /home/scardoso/temp/R-IL308C2/R-IL308C2/10/pdata/1
## Reading sample  R-IL308C3  in  /home/scardoso/temp/R-IL308C3/R-IL308C3/10/pdata/1
## Reading sample  R-IL308C4  in  /home/scardoso/temp/R-IL308C4/R-IL308C4/10/pdata/1
## Reading sample  R-IL308C8  in  /home/scardoso/temp/R-IL308C8/R-IL308C8/10/pdata/1
## Reading sample  R-IL308T1  in  /home/scardoso/temp/R-IL308T1/R-IL308T1/10/pdata/1
## Reading sample  R-IL308T2  in  /home/scardoso/temp/R-IL308T2/R-IL308T2/10/pdata/1
## Reading sample  R-IL308T3  in  /home/scardoso/temp/R-IL308T3/R-IL308T3/10/pdata/1
## Reading sample  R-IL308T4  in  /home/scardoso/temp/R-IL308T4/R-IL308T4/10/pdata/1
## Reading sample  R-IL308T8  in  /home/scardoso/temp/R-IL308T8/R-IL308T8/10/pdata/1
## Reading sample  R-IL7C1  in  /home/scardoso/temp/R-IL7C1/R-IL7C1/10/pdata/1
## Reading sample  R-IL7C2  in  /home/scardoso/temp/R-IL7C2/R-IL7C2/10/pdata/1
## Reading sample  R-IL7C3  in  /home/scardoso/temp/R-IL7C3/R-IL7C3/10/pdata/1
## Reading sample  R-IL7C4  in  /home/scardoso/temp/R-IL7C4/R-IL7C4/10/pdata/1
## Reading sample  R-IL7C8  in  /home/scardoso/temp/R-IL7C8/R-IL7C8/10/pdata/1
## Reading sample  R-IL7T1  in  /home/scardoso/temp/R-IL7T1/R-IL7T1/10/pdata/1
## Reading sample  R-IL7T2  in  /home/scardoso/temp/R-IL7T2/R-IL7T2/10/pdata/1
## Reading sample  R-IL7T3  in  /home/scardoso/temp/R-IL7T3/R-IL7T3/10/pdata/1
## Reading sample  R-IL7T4  in  /home/scardoso/temp/R-IL7T4/R-IL7T4/10/pdata/1
## Reading sample  R-IL7T8  in  /home/scardoso/temp/R-IL7T8/R-IL7T8/10/pdata/1
## Reading sample  R-KDC1  in  /home/scardoso/temp/R-KDC1/R-KDC1/10/pdata/1
## Reading sample  R-KDC2  in  /home/scardoso/temp/R-KDC2/R-KDC2/10/pdata/1
## Reading sample  R-KDC3  in  /home/scardoso/temp/R-KDC3/R-KDC3/10/pdata/1
## Reading sample  R-KDC4  in  /home/scardoso/temp/R-KDC4/R-KDC4/10/pdata/1
## Reading sample  R-KDC8  in  /home/scardoso/temp/R-KDC8/R-KDC8/10/pdata/1
## Reading sample  R-KDT1  in  /home/scardoso/temp/R-KDT1/R-KDT1/10/pdata/1
## Reading sample  R-KDT2  in  /home/scardoso/temp/R-KDT2/R-KDT2/10/pdata/1
## Reading sample  R-KDT3  in  /home/scardoso/temp/R-KDT3/R-KDT3/10/pdata/1
## Reading sample  R-KDT4  in  /home/scardoso/temp/R-KDT4/R-KDT4/10/pdata/1
## Reading sample  R-KDT8  in  /home/scardoso/temp/R-KDT8/R-KDT8/10/pdata/1
## Creating dataset (this may take a while)
## Done.
```

3. Table of the data just loaded:

Hide

```
DT::datatable(nmr_bruker_spectra_dataset$data, options=list(scrollX = TRUE))
```

```
## Warning in instance$preRenderHook(instance): It seems your data is too big for client-side DataTables. You may consider server-side
processing:
## https://rstudio.github.io/DT/server.html
```

4. Table of the metadata just loaded:

Hide

```
DT::datatable(nmr_bruker_spectra_dataset$metadata)
```

5. Detection and alignment of peaks:

Here, we want the detected peaks to have a minimum intensity of 5000000. They will also be aligned according to the specmine method, with a step value of 0.03. But first, we have to treat the missing values (details below), otherwize the function to detect and align peaks will not work.

Hide

```
#Treat missing values:
nmr_bruker_spectra_dataset_mv=missingvalues_imputation(nmr_bruker_spectra_dataset, method = "value", value=0)

#Detect and align peaks, now that missing values were treated:
nmr_bruker_peaks_dataset=detect_nmr_peaks_from_dataset(nmr_bruker_spectra_dataset_mv, baseline_tresh = 5000000)
```

```
## Spectrum  L-IL308C1-1  has  42  peaks.
## Spectrum  L-IL308C1-2  has  10  peaks.
## Spectrum  L-IL308C1-3  has  8  peaks.
## Spectrum  L-IL308C2-1  has  13  peaks.
## Spectrum  L-IL308C2-2  has  21  peaks.
## Spectrum  L-IL308C2-3  has  7  peaks.
## Spectrum  L-IL308C3-1  has  42  peaks.
```

```
## Spectrum  L-IL308C3-2  has  5  peaks.
## Spectrum  L-IL308C3-3  has  6  peaks.
## Spectrum  L-IL308C4-1  has  44  peaks.
## Spectrum  L-IL308C4-2  has  16  peaks.
## Spectrum  L-IL308C4-3  has  15  peaks.
## Spectrum  L-IL308C8-1  has  23  peaks.
## Spectrum  L-IL308C8-2  has  7  peaks.
## Spectrum  L-IL308C8-3  has  13  peaks.
## Spectrum  L-IL308T1-1  has  39  peaks.
## Spectrum  L-IL308T1-2  has  23  peaks.
## Spectrum  L-IL308T1-3  has  9  peaks.
## Spectrum  L-IL308T2-1  has  45  peaks.
## Spectrum  L-IL308T2-2  has  14  peaks.
## Spectrum  L-IL308T2-3  has  4  peaks.
## Spectrum  L-IL308T3-1  has  33  peaks.
## Spectrum  L-IL308T3-2  has  11  peaks.
## Spectrum  L-IL308T3-3  has  6  peaks.
## Spectrum  L-IL308T4-1  has  26  peaks.
## Spectrum  L-IL308T4-2  has  6  peaks.
## Spectrum  L-IL308T4-3  has  5  peaks.
## Spectrum  L-IL308T8-1  has  40  peaks.
## Spectrum  L-IL308T8-2  has  17  peaks.
## Spectrum  L-IL308T8-3  has  17  peaks.
## Spectrum  L-IL7C1-1  has  29  peaks.
## Spectrum  L-IL7C1-2  has  10  peaks.
## Spectrum  L-IL7C1-3  has  6  peaks.
## Spectrum  L-IL7C2-1  has  34  peaks.
## Spectrum  L-IL7C2-2  has  11  peaks.
## Spectrum  L-IL7C2-3  has  8  peaks.
## Spectrum  L-IL7C3-1  has  36  peaks.
## Spectrum  L-IL7C3-2  has  8  peaks.
## Spectrum  L-IL7C3-3  has  8  peaks.
## Spectrum  L-IL7C4-1  has  59  peaks.
## Spectrum  L-IL7C4-2  has  9  peaks.
## Spectrum  L-IL7C4-3  has  6  peaks.
## Spectrum  L-IL7C8-1  has  24  peaks.
## Spectrum  L-IL7C8-2  has  11  peaks.
## Spectrum  L-IL7C8-3  has  7  peaks.
## Spectrum  L-IL7T1-1  has  47  peaks.
## Spectrum  L-IL7T1-2  has  6  peaks.
## Spectrum  L-IL7T1-3  has  11  peaks.
## Spectrum  L-IL7T2-1  has  26  peaks.
## Spectrum  L-IL7T2-2  has  6  peaks.
## Spectrum  L-IL7T2-3  has  7  peaks.
## Spectrum  L-IL7T3-1  has  23  peaks.
## Spectrum  L-IL7T3-2  has  10  peaks.
## Spectrum  L-IL7T3-3  has  11  peaks.
## Spectrum  L-IL7T4-1  has  29  peaks.
## Spectrum  L-IL7T4-2  has  9  peaks.
## Spectrum  L-IL7T4-3  has  7  peaks.
## Spectrum  L-IL7T8-1  has  16  peaks.
## Spectrum  L-IL7T8-2  has  8  peaks.
## Spectrum  L-IL7T8-3  has  12  peaks.
## Spectrum  L-KDC1-1  has  40  peaks.
## Spectrum  L-KDC1-2  has  5  peaks.
## Spectrum  L-KDC1-3  has  5  peaks.
## Spectrum  L-KDC2-1  has  42  peaks.
## Spectrum  L-KDC2-2  has  4  peaks.
## Spectrum  L-KDC2-3  has  6  peaks.
## Spectrum  L-KDC3-1  has  44  peaks.
## Spectrum  L-KDC3-2  has  14  peaks.
## Spectrum  L-KDC3-3  has  8  peaks.
## Spectrum  L-KDC4-1  has  30  peaks.
## Spectrum  L-KDC4-2  has  14  peaks.
## Spectrum  L-KDC4-3  has  4  peaks.
## Spectrum  L-KDC8-1  has  29  peaks.
## Spectrum  L-KDC8-2  has  11  peaks.
## Spectrum  L-KDC8-3  has  12  peaks.
## Spectrum  L-KDT1-1  has  44  peaks.
## Spectrum  L-KDT1-2  has  10  peaks.
## Spectrum  L-KDT1-3  has  6  peaks.
## Spectrum  L-KDT2-1  has  35  peaks.
## Spectrum  L-KDT2-2  has  4  peaks.
## Spectrum  L-KDT2-3  has  5  peaks.
## Spectrum  L-KDT3-1  has  14  peaks.
## Spectrum  L-KDT3-2  has  13  peaks.
## Spectrum  L-KDT3-3  has  6  peaks.
## Spectrum  L-KDT4-1  has  33  peaks.
## Spectrum  L-KDT4-2  has  7  peaks.
## Spectrum  L-KDT4-3  has  5  peaks.
## Spectrum  L-KDT8-1  has  13  peaks.
## Spectrum  L-KDT8-2  has  3  peaks.
## Spectrum  L-KDT8-3  has  4  peaks.
## Spectrum  R-IL308C1  has  13  peaks.
## Spectrum  R-IL308C2  has  25  peaks.
## Spectrum  R-IL308C3  has  22  peaks.
## Spectrum  R-IL308C4  has  21  peaks.
## Spectrum  R-IL308C8  has  22  peaks.
## Spectrum  R-IL308T1  has  28  peaks.
## Spectrum  R-IL308T2  has  19  peaks.
## Spectrum  R-IL308T3  has  19  peaks.
## Spectrum  R-IL308T4  has  27  peaks.
## Spectrum  R-IL308T8  has  15  peaks.
## Spectrum  R-IL7C1  has  7  peaks.
## Spectrum  R-IL7C2  has  26  peaks.
```

```
## Spectrum  R-IL7C3  has  19  peaks.
## Spectrum  R-IL7C4  has  38  peaks.
## Spectrum  R-IL7C8  has  22  peaks.
## Spectrum  R-IL7T1  has  23  peaks.
## Spectrum  R-IL7T2  has  15  peaks.
## Spectrum  R-IL7T3  has  21  peaks.
## Spectrum  R-IL7T4  has  10  peaks.
## Spectrum  R-IL7T8  has  12  peaks.
## Spectrum  R-KDC1  has  26  peaks.
## Spectrum  R-KDC2  has  11  peaks.
## Spectrum  R-KDC3  has  24  peaks.
## Spectrum  R-KDC4  has  14  peaks.
## Spectrum  R-KDC8  has  17  peaks.
## Spectrum  R-KDT1  has  25  peaks.
## Spectrum  R-KDT2  has  21  peaks.
## Spectrum  R-KDT3  has  17  peaks.
## Spectrum  R-KDT4  has  9  peaks.
## Spectrum  R-KDT8  has  4  peaks.
```

6. Table of the data with the peaks detected:

Hide

```
DT::datatable(nmr_bruker_peaks_dataset$data, options=list(scrollX = TRUE))
```

4.3.2.2 VARIAN data

The example data here used was obtained from the Metabolights database, under the ID MTBLS346. The data was obtained using the *get_metabolights_study* function.

The folder named "14" was not used in this example (i.e. not present in the data folder here provided), as data was too big to be read. Therefore, its metadata information was also deleted from the metadata file.

1. Strings indicating where data and metadata is:

Hide

```
varian_nmr_spectra_folder="/home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346"
varian_nmr_metadata_file="/home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/metadata_no_14.csv"
```

2. In this data, the spectra folders' names do not correspond to the names of the samples, so a file (here represented by variable *samples_folders*) will be given to the argument samples.names.

Hide

```
samples_folders="/home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/samples_files.csv"
```

Content of *samples_folders* file, where the first column corresponds to the names of the samples and the second one to the folders names where the corresponding data is stored:

```
THS-WT1,1.fid.zip
THS-WT2,2.fid.zip
THS-WT3,3.fid.zip
THS-WT4,4.fid.zip
THS-WT5,5.fid.zip
THS-WT6,6.fid.zip
THS-WT7,7.fid.zip
THS-WT8,8.fid.zip
THS-WT9,9.fid.zip
THS-GM1_1,10.fid.zip
THS-GM1_2,11.fid.zip
THS-GM1_3,12.fid.zip
THS-GM1_4,13.fid.zip
THS-GM1_5,14.fid.zip
THS-GM1_6,15.fid.zip
THS-GM1_7,16.fid.zip
THS-GM1_8,17.fid.zip
THS-GM1_9,18.fid.zip
SIL-WT1,19.fid.zip
SIL-WT2,20.fid.zip
SIL-WT3,21.fid.zip
SIL-WT4,22.fid.zip
SIL-WT5,23.fid.zip
SIL-WT6,24.fid.zip
SIL-WT7,25.fid.zip
SIL-WT8,26.fid.zip
SIL-WT9,27.fid.zip
SIL-GM1_1,28.fid.zip
SIL-GM1_2,29.fid.zip
SIL-GM1_3,30.fid.zip
SIL-GM1_4,31.fid.zip
SIL-GM1_5,32.fid.zip
SIL-GM1_6,33.fid.zip
SIL-GM1_7,34.fid.zip
SIL-GM1_8,35.fid.zip
SIL-GM1_9,36.fid.zip
SIL-GM2_1,37.fid.zip
SIL-GM2_2,38.fid.zip
SIL-GM2_3,39.fid.zip
SIL-GM2_4,40.fid.zip
SIL-GM2_5,41.fid.zip
SIL-GM2_6,42.fid.zip
SIL-GM2_7,43.fid.zip
SIL-GM2_8,44.fid.zip
SIL-GM2_9,45.fid.zip
```

2. Loading NMR vartian spectral data to specmine:

The folders here read are not zipped (.zip), as they previously uncompressed for purpose of showing how to proceed in such a case. No zero filling will be performed, but apodization will be

Hide

```
nmr_varian_spectra_dataset=read_varian_spectra_raw(varian_nmr_spectra_folder, metadata_file=varian_nmr_metadata_file, samples.names=sa
mples_folders, zipped=F,
                                                   zero_filling=F)
```

```
## Reading files:
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/10.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/11.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/12.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/13.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/15.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/16.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/17.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/18.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/19.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/1.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/20.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/21.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/22.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/23.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/24.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/25.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/26.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/27.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/28.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/29.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/2.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/30.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/31.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/32.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/33.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/34.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/35.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/36.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/37.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/38.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/39.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/3.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/40.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/41.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/42.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/43.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/44.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/45.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/4.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/5.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/6.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/7.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/8.fid
## /home/scardoso/Documents/metabolomics_datasets/NMR/Spectra/MTBLS346/9.fid
```

3. Table of the first 500 data points of the data just loaded:

Hide

```
DT::datatable(nmr_varian_spectra_dataset$data[1:500,], options=list(scrollX = TRUE))
```

4. Table of the metadata just loaded:

Hide

```
DT::datatable(nmr_varian_spectra_dataset$metadata)
```

5. Detection and alignment of peaks:

Here, we want the detected peaks to have a minimum intensity of 20000000. They will also be aligned according to the specmine method, with a step value of 0.03.

Hide

```
#Detect and align peaks:
nmr_varian_peaks_dataset=detect_nmr_peaks_from_dataset(nmr_varian_spectra_dataset, baseline_tresh = 20000000)
```

```
## Spectrum  THS-GM1_1  has  33  peaks.
## Spectrum  THS-GM1_2  has  33  peaks.
## Spectrum  THS-GM1_3  has  34  peaks.
## Spectrum  THS-GM1_4  has  34  peaks.
## Spectrum  THS-GM1_6  has  32  peaks.
## Spectrum  THS-GM1_7  has  32  peaks.
## Spectrum  THS-GM1_8  has  33  peaks.
## Spectrum  THS-GM1_9  has  32  peaks.
## Spectrum  SIL-WT1  has  35  peaks.
## Spectrum  THS-WT1  has  34  peaks.
## Spectrum  SIL-WT2  has  32  peaks.
## Spectrum  SIL-WT3  has  32  peaks.
## Spectrum  SIL-WT4  has  33  peaks.
## Spectrum  SIL-WT5  has  33  peaks.
## Spectrum  SIL-WT6  has  32  peaks.
## Spectrum  SIL-WT7  has  32  peaks.
## Spectrum  SIL-WT8  has  32  peaks.
## Spectrum  SIL-WT9  has  31  peaks.
## Spectrum  SIL-GM1_1  has  42  peaks.
## Spectrum  SIL-GM1_2  has  32  peaks.
## Spectrum  THS-WT2  has  35  peaks.
## Spectrum  SIL-GM1_3  has  33  peaks.
## Spectrum  SIL-GM1_4  has  65  peaks.
## Spectrum  SIL-GM1_5  has  32  peaks.
## Spectrum  SIL-GM1_6  has  31  peaks.
## Spectrum  SIL-GM1_7  has  30  peaks.
## Spectrum  SIL-GM1_8  has  33  peaks.
## Spectrum  SIL-GM1_9  has  30  peaks.
## Spectrum  SIL-GM2_1  has  31  peaks.
## Spectrum  SIL-GM2_2  has  43  peaks.
## Spectrum  SIL-GM2_3  has  32  peaks.
## Spectrum  THS-WT3  has  36  peaks.
## Spectrum  SIL-GM2_4  has  34  peaks.
## Spectrum  SIL-GM2_5  has  36  peaks.
## Spectrum  SIL-GM2_6  has  33  peaks.
## Spectrum  SIL-GM2_7  has  32  peaks.
## Spectrum  SIL-GM2_8  has  52  peaks.
## Spectrum  SIL-GM2_9  has  31  peaks.
## Spectrum  THS-WT4  has  34  peaks.
## Spectrum  THS-WT5  has  32  peaks.
## Spectrum  THS-WT6  has  33  peaks.
## Spectrum  THS-WT7  has  32  peaks.
## Spectrum  THS-WT8  has  32  peaks.
## Spectrum  THS-WT9  has  34  peaks.
```

6. Table of the data with the peaks detected:

Hide

```
DT::datatable(nmr_varian_peaks_dataset$data, options=list(scrollX = TRUE))
```

4.4 GC/LC-MS Spectra (peak detection comes with data reading)

4.4.1 Functions to use

Read MS spectra data and metadata into specmine dataset

> read_ms_spectra(folder.name, type = “undefined”,filename.meta = NULL, description = “”, prof.method = “bin”, fwhm = 30, bw = 30, intvalue = “into”, header.col.meta = TRUE, header.row.meta = TRUE, sep.meta = “,”)

- *folder.name*: string containing the path of the data folder;
- *type*: type of the data. Defaults to “undefined”, but in this case should either be “lcms-spectra” or “gcms-spectra”;
- *filename.meta*: string containing the path of the metadata file; **optional but recomended**
- *description*: a short text describing the dataset. **optional**
- *prof.method*: profile generation method. Either “bin”, “binlin”, “binlinbase” or “intlin”. Defaults to “bin”. profmethod parameter from xcmsSet function from xcms package.
- *fwhm*: full width at half maximum of matched filtration gaussian model peak. Commonly 30 for LC-MS spectra and 4 for GC-MS spectra. Defaults to 30. fwhm parameter from xcmsSet function from xcms package.
- *bw*: bandwidth (standard deviation or half width at half maximum) of the Gaussian smoothing kernel, to apply to the peak density chromatogram. Commonly 30 for LC-MS spectra and 5 for GC-MS spectra. Defaults to 30. bw parameter from group function from xcms package.
- *intvalue*: peak intensity measure (intvalue value parameter from groupval function from xcms package). It can either be
  - “into” - integrated area of original (raw) peak. Default value;
  - “intf” - integrated area of filtered peak;
  - “maxo” - maximum intensity of original (raw) peak;
  - “maxf” - maximum intensity of filtered peak;
- *header.col.meta* : boolean value (TRUE or FALSE) indicating if the metadata file contains a header column with the name of the metadata variables. Defaults to TRUE.
- *header.row.meta*: boolean value (TRUE or FALSE) indicating if the metadata file contains a header row with the name of the samples. Defaults to TRUE.
- *sep.meta*: the separator character of the metadata file. Defaults to “,”.

4.4.2 Example

1. Strings indicating where data and metadata is:

Hide

```
lcms_data_folder="/home/scardoso/Documents/metabolomics_datasets/MS/Spectra/miceSpinalCord/data"
lcms_metadata_file="/home/scardoso/Documents/metabolomics_datasets/MS/Spectra/miceSpinalCord/metadata_lcms.csv"
```

2. Loading LC-MS data to specmine:

Hide

```
lcms_dataset = read_ms_spectra(lcms_data_folder, type = "lcms-spectra", lcms_metadata_file, description = "lc-ms mice spinal cord samples")
```

```
## Processing 3195 mz slices ... OK
## Performing retention time correction using 133 peak groups.
## Processing 3195 mz slices ... OK
## /home/scardoso/Documents/metabolomics_datasets/MS/Spectra/miceSpinalCord/data/ko15.CDF
## method:  bin
## step:  0.1
## /home/scardoso/Documents/metabolomics_datasets/MS/Spectra/miceSpinalCord/data/ko16.CDF
## method:  bin
## step:  0.1
## /home/scardoso/Documents/metabolomics_datasets/MS/Spectra/miceSpinalCord/data/ko21.CDF
## method:  bin
## step:  0.1
## /home/scardoso/Documents/metabolomics_datasets/MS/Spectra/miceSpinalCord/data/ko22.CDF
## method:  bin
## step:  0.1
## /home/scardoso/Documents/metabolomics_datasets/MS/Spectra/miceSpinalCord/data/wt21.CDF
## method:  bin
## step:  0.1
## /home/scardoso/Documents/metabolomics_datasets/MS/Spectra/miceSpinalCord/data/wt22.CDF
## method:  bin
## step:  0.1
## /home/scardoso/Documents/metabolomics_datasets/MS/Spectra/miceSpinalCord/data/wt15.CDF
## method:  bin
## step:  0.1
## /home/scardoso/Documents/metabolomics_datasets/MS/Spectra/miceSpinalCord/data/wt16.CDF
## method:  bin
## step:  0.1
## /home/scardoso/Documents/metabolomics_datasets/MS/Spectra/miceSpinalCord/data/ko18.CDF
## method:  bin
## step:  0.1
## /home/scardoso/Documents/metabolomics_datasets/MS/Spectra/miceSpinalCord/data/ko19.CDF
## method:  bin
## step:  0.1
## /home/scardoso/Documents/metabolomics_datasets/MS/Spectra/miceSpinalCord/data/wt18.CDF
## method:  bin
## step:  0.1
## /home/scardoso/Documents/metabolomics_datasets/MS/Spectra/miceSpinalCord/data/wt19.CDF
## method:  bin
## step:  0.1
```

3. Table of the data just loaded:

Hide

```
DT::datatable(lcms_dataset$data, options=list(scrollX = TRUE))
```

4. Table of the metadata just loaded:

Hide

```
DT::datatable(lcms_dataset$metadata)
```

4.5 UV-Vis, IR and Raman Spectra

4.5.1 Functions to use

4.5.1.1 When the data is in a CSV file

Read data and metadata

> read_dataset_csv(filename.data, filename.meta = NULL, type = "undefined", description = "", label.x = NULL, label.values = NULL, sample.names = NULL, format = "row", header.col = TRUE, header.row = TRUE, sep = ",", header.col.meta = TRUE, header.row.meta = TRUE, sep.meta = ",")

- *filename.data*: string containing the path of the data file;
- *filename.meta*: string containing the path metadata file; **optional but recomended**
- *type*: type of the data. Defaults to "undefined". It should be "concentrations";
- *description*: a short text describing the dataset. **optional**
- *label.x*: the label for the x values. **optional**
- *label.values*: the label for the y values. **optional**
- *sample.names*: character vector with the names of the samples, if they are not present in the header row or column;
- *format*: format which the data are in the CSV file. It can be
  - "row": if the samples are in the rows (each line corresponds to a sample);
  - "col" if the samples are in the columns (each column corresponds to a sample).
- *header.col*: boolean value (TRUE or FALSE) indicating if the CSV contains a header column with the names of the samples (if format="row") or variables (if format="col"). Defaults to TRUE;

- *header.row*: boolean value (TRUE or FALSE) indicating if the CSV contains a header row with the names of the variables (if format="row") or samples (if format="col"). Defaults to TRUE;
- *sep*: the separator character. Defaults to ",";
- *header.col.meta*: boolean value indicating if the metadata CSV file contains a header column with the name of the metadata variables. Defaults to TRUE;
- *header.row.meta*: boolean value indicating if the metadata CSV file contains a header row with the name of the samples. Defaults to TRUE;
- *sep.meta*: the separator character of the metadata file. Defaults to ",".

4.5.1.2 When the data is in multiple CSV files

1: Read Data into list of data samples

```
read_csvs_folder(foldername, header=TRUE, sep=",", dec=".", ...)
```

- *foldername*: string containing the path of the data folder;
- *header*: boolean value (TRUE or FALSE) indicating whether data files have a header row with the names of the data variables. Defaults to TRUE;
- *sep*: the separator character of the data values. Defaults to ",";
- *dec*: character used in the file for decimal points. Defaults to ".";
- *skip*: number of lines of the data file to skip before beginning to read data; **optional**
- ...: additional parameters for read.csv function from utils package.

2: Read metadata file (optional step but recomended)

```
read_metadata(filename, header.col = T, header.row = T, sep = ",")
```

- *filename*: string indicating the path of the file with the metadata;
- *header.col*: boolean value (TRUE or FALSE) indicating if the metadata CSV file contains a header column with the name of the metadata variables. Defaults to TRUE;
- *header.row*: boolean value (TRUE or FALSE) indicating if the metadata CSV file contains a header row with the name of the samples. Defaults to TRUE;
- *sep*: the separator character. Defaults to ",".

3: Join Data and Metadata (adition of metadata is, as said, optional but recomended) into a specmine dataset

```
create_dataset(datamatrix, type = "undefined", metadata = NULL, description = "", sample.names = NULL, x.axis.values = NULL, label.x = NULL, label.values = NULL)
```

- *datamatrix*: matrix with numerical data: rows are assumed to be variables and columns assumed to be samples. Can be obatined from the function *read_csvs_folder* above, by transforming the data list obtained into a numeric matrix. See examples for this;
- *type*: type of the data. Defaults to "undefined", but in this case should either be "ir-spectra", "uvv-spectra" or "raman-spectra";
- *metadata*: data frame containing the metadata. Can be obtained from the function *read_metadata* above; **optional but recomended**
- *description*: string with a short description of the dataset; **optional**
- *sample.names*: vector with sample names. If NULL then the column names of datamatrix or sequential numbers will be used. Defaults to NULL;
- *x.axis.values*: vector with the x axis values. If NULL then the row names of datamatrix or sequential numbers will be used. Defaults to NULL
- *label.x*: the label for the x values. **optional**
- *label.y*: the label for the y values. **optional**

4.5.1.3 When the data is in multiple SPC files

Read Data and Metadata

```
read_dataset_spc(folder.data, filename.meta = NULL, type = "undefined", description = "", nosubhdr = F, label.x = NULL, label.values = NULL, header.col.meta = TRUE, header.row.meta = TRUE, sep.meta = ",")
```

- *folder.data*: string containing the path of the data folder;
- *filename.meta*: string indicating the path of the file with the metadata;
- *type*: type of the data. Defaults to "undefined", but in this case should either be "ir-spectra", "uvv-spectra" or "raman-spectra";
- *description*: string with a short description of the dataset; **optional**
- *nosubhdr*: boolean value (TRUE or FALSE) indicating if the subheader of the file should be read or not. Defaults to FALSE;
- *label.x*: the label for the x values. **optional**
- *label.values*: the label for the y values. **optional**
- *header.col.meta*: boolean value (TRUE or FALSE) indicating if the metadata CSV file contains a header column with the name of the metadata variables. Defaults to TRUE;
- *header.row.meta*: boolean value (TRUE or FALSE) indicating if the metadata CSV file contains a header row with the name of the samples. Defaults to TRUE;
- *sep.meta*: the separator character of the metadata file. Defaults to ",".

4.5.1.4 When the data is in multiple (J)DX files:

Read Data and Metadata

```
read_dataset_dx(folder.data, filename.meta = NULL, type = "undefined", description = "", label.x = NULL, label.values = NULL, header.col.meta = TRUE, header.row.meta = TRUE, sep.meta = ",")
```

- *folder.data*: string containing the path of the data folder;
- *filename.data*: string containing the path of the filename; **optional but recomended**
- *type*: type of data. Defaults to "undefined", but in this case should either be "ir-spectra", "uvv-spectra" or "raman-spectra";
- *description*: a short text describing the dataset; **optional**
- *label.x*: the label for the x values; **optional**
- *label.values*: the label for the y values; **optional**
- *header.col.meta*: boolean value (TRUE or FALSE) indicating if the metadata CSV file contains a header column with the name of the metadata variables. Defaults to TRUE;

- *header.row.meta*: boolean value (TRUE or FALSE) indicating if the metadata CSV file contains a header row with the name of the samples. Defaults to TRUE;
- *sep.meta*: the separator character of the metadata file. Defaults to ",".

4.5.1.5 When the data is in multiple XLSX files:

To read XLSX data files into specmine, the function *read.xlsx* must be used to read each data file. See example for a better understanding of the overall steps to perform.

1: Read Data (function from xlsx paxkage)

Only the more imnportant arguments for the case of metabolomics spectra will be explained bellow. For further information, go here.

> read.xlsx(file, sheetIndex,header=TRUE, ...)

- *file*: string containing the path of one of the data files;
- *sheetIndex*: a number representing the sheet index in the workbook. Normally should be 1 (the first sheet in the workbook);
- *header*: a boolean value (TRUE or FALSE) indicating whether the first row contains the names of the columns.

2: Read metadata file (optional step but recomended)

> read_metadata(filename, header.col = T, header.row = T, sep = ",")

- *filename*: string indicating the path of the file with the metadata;
- *header.col*: boolean value (TRUE or FALSE) indicating if the metadata CSV file contains a header column with the name of the metadata variables. Defaults to TRUE;
- *header.row*: boolean value (TRUE or FALSE) indicating if the metadata CSV file contains a header row with the name of the samples. Defaults to TRUE;
- *sep*: the separator character. Defaults to ",".

3: Join Data and Metadata (adition of metadata is, as said, optional but recomended) into a specmine dataset

> create_dataset(datamatrix, type = "undefined", metadata = NULL, description = "", sample.names = NULL, x.axis.values = NULL, label.x = NULL, label.values = NULL)

- *datamatrix*: matrix with numerical data: rows are assumed to be variables and columns assumed to be samples. Can be obatined from the function *read_csvs_folder* above, by transforming the data list obtained into a numeric matrix. See examples for this;
- *type*: type of the data. Defaults to "undefined", but in this case should either be "ir-spectra", "uvv-spectra" or "raman-spectra";
- *metadata*: data frame containing the metadata. Can be obtained from the function *read_metadata* above; **optional but recomended**
- *description*: string with a short description of the dataset; **optional**
- *sample.names*: vector with sample names. If NULL then the column names of datamatrix or sequential numbers will be used. Defaults to NULL;
- *x.axis.values*: vector with the x axis values. If NULL then the row names of datamatrix or sequential numbers will be used. Defaults to NULL
- *label.x*: the label for the x values. **optional**
- *label.y*: the label for the y values. **optional**

4.5.2 Example(s)

4.5.2.1 Data in CSV File (IR Spectra)

1: Strings indicating where data and metadata is:

Hide

```
ir_spectra_data_file="/home/scardoso/Downloads/cassavaPPD/data_cassava.csv"
ir_spectra_metadata_file="/home/scardoso/Downloads/cassavaPPD/metadata_ir.csv"
```

2: Read data and metadata files:

Hide

```
ir_spectra_dataset=read_dataset_csv(ir_spectra_data_file, ir_spectra_metadata_file, type = "ir-spectra", format = "col")
```

3. Table of the data just loaded:

Hide

```
DT::datatable(ir_spectra_dataset$data[1:100,], options=list(scrollX = TRUE))
```

4. Table of the metadata just loaded:

Hide

```
DT::datatable(ir_spectra_dataset$metadata)
```

4.5.2.2 XLSX Files (UV-Vis Spectra)

1: Strings indicating where data and metadata is:

Hide

```
uvvis_spectra_data_folder="/home/scardoso/Downloads/Cassava Carotenoids/data"
uvvis_spectra_metadata_file="/home/scardoso/Downloads/Cassava Carotenoids/metadata_uvvis.csv"
```

2: First, we have to know the number of rows in the files, which are the number of data points, equal to all XLSX files:

Hide

```
#List of file paths in the data folder:
files = list.files(uvvis_spectra_data_folder, full.names = T)

#Names of the files in the data folder:
fileNames = list.files(uvvis_spectra_data_folder)

#Read one of the files and get the number of rows and the values of the data points:
file = xlsx::read.xlsx(files[1], sheetIndex = 1, header=F)
nrows = nrow(file) #number of rows
rnames = file[,1] #Values of the data points
```

3: Prepare data matrix to read to there the spectra data:

Hide

```
#Data matrix:
datamat = matrix(nrow = nrows, ncol = length(files))

#Rownames of the data matrix (values of data points):
rownames(datamat) = rnames

#Colnames of data matrix (samples names). Here, it is considered that the samples names are the names given to the files:
ext = paste('.', tools::file_ext(fileNames[1]), sep = '') #Gets the extension of the data files, so that it is taken away from the fil
e name and only
                                                    #the name of the file is inserted
colnames(datamat) = gsub(ext, "", fileNames)
```

4: Read data files into data matrix:

Hide

```
for (i in 1:length(files)){
  tab_excel = xlsx::read.xlsx(files[i], sheetIndex = 1, header = F)
  datamat[,i] = c(tab_excel[,2], rep(NA, nrows-length(tab_excel[,2]))) #If there's missing data
}
```

5: Read metadata file:

Hide

```
uvvis_spectra_metadata=read_metadata(uvvis_spectra_metadata_file)
```

6: Join data and metadata into the specmine dataset:

Hide

```
uvvis_spectra_dataset=create_dataset(datamat, uvvis_spectra_metadata, type = "uvv-spectra")
```

7. Table of the data just loaded:

Hide

```
DT::datatable(uvvis_spectra_dataset$data[1:100,], options=list(scrollX = TRUE))
```

8. Table of the metadata just loaded:

Hide

```
DT::datatable(uvvis_spectra_dataset$metadata, options=list(scrollX = TRUE))
```

4.6 Concentrations Data

4.6.1 Functions to use

Read data and metadata

> read_dataset_csv(filename.data, filename.meta = NULL, type = “undefined”, description = “”, label.x = NULL, label.values = NULL, sample.names = NULL, format = “row”, header.col = TRUE, header.row = TRUE, sep = “,”, header.col.meta = TRUE, header.row.meta = TRUE, sep.meta = “,”)

- *filename.data*: string containing the path of the data file;
- *filename.meta*: string containing the path metadata file; **optional but recomended**
- *type*: type of the data. Defaults to “undefined”. It should be “concentrations”;
- *description*: a short text describing the dataset. **optional**
- *label.x*: the label for the x values. **optional**
- *label.values*: the label for the y values. **optional**
- *sample.names*: character vector with the names of the samples, if they are not present in the header row or column;
- *format*: format which the data are in the CSV file. It can be
  - “row”: if the samples are in the rows (each line corresponds to a sample);
  - “col” if the samples are in the columns (each column corresponds to a sample).
- *header.col*: boolean value (TRUE or FALSE) indicating if the CSV contains a header column with the names of the samples (if format=“row”) or variables (if format=“col”). Defaults to TRUE;
- *header.row*: boolean value (TRUE or FALSE) indicating if the CSV contains a header row with the names of the variables (if format=“row”) or samples (if format=“col”). Defaults to TRUE;
- *sep*: the separator character. Defaults to “,”;

- *header.col.meta*: boolean value indicating if the metadata CSV file contains a header column with the name of the metadata variables. Defaults to TRUE;
- *header.row.meta*: boolean value indicating if the metadata CSV file contains a header row with the name of the samples. Defaults to TRUE;
- *sep.meta*: the separator character of the metadata file. Defaults to ",".

4.6.2 Example

1. Strings indicating where data and metadata is:

Hide

```
concentrations_data_file="/home/scardoso/Documents/metabolomics_datasets/concentrations/cachexia/human_cachexia.csv"
concentrations_metadata_file="/home/scardoso/Documents/metabolomics_datasets/concentrations/cachexia/metadata_concentrations.csv"
```

2. Read concentrations data file and metadata file:

Hide

```
concentrations_dataset=read_dataset_csv(concentrations_data_file, filename.meta = concentrations_metadata_file, type = "concentrations", description = "Metabolite concentrations from samples with or without cachexia", label.x = "metabolites", label.values = "concentration", format = "row")
```

3. Table of the data just loaded:

Hide

```
DT::datatable(concentrations_dataset$data, options=list(scrollX = TRUE))
```

4. Table of the metadata just loaded:

Hide

```
DT::datatable(concentrations_dataset$metadata)
```

5 Upload Data from Metabolights Database

5.1 Function to use

TO USE THIS FUNCTION YOU MUST HAVE PYTHON3 INSTALLED, WITH THE MODULES isatools, os, ftplib, glob, logging, pandas, tempfile, shutil and re INSTALLED

> get_metabolights_study(mtblsID, directory)

- *mtblsID*: string indicating the ID of the metabolights study to download;
- *directory*: string indicating the directory to which the data files of the study will be downloaded. The data will be stored in "/directory/mtblsID".

5.2 Example

Hide

```
get_metabolights_study("MTBLS131", "/home/scardoso/Documents")
```

```
## Downloading files... please wait
##
## Done.
```

6 Data Visualization

6.1 Dataset Summary

This function provides a summary of the main features of the dataset:

- Description;
- Typr of data;
- Number of samples;
- Number of datapoints;
- Number of metadata variables, if there is metadata;
- Labels of x axis values and data points, if labels were given.
- If the user wants to know some statistical properties:
    - Number of missing values in data;
    - Mean of data values;
    - Median of data values;
    - Standard deviation;
    - Range of values;
    - Quantiles;

6.1.1 Function to use

> sum_dataset(dataset, stats = T)

- *dataset*: a specmine dataset;
- *stats*: boolean value (TRUE or FALSE) indicating whether some global statistics of the data values should be calculated. Defaults to TRUE

6.1.2 Example

Example makes use of an LC-MS dataset (LC-MS dataset used).

1. Data summary with statistics:

Hide

```
sum_dataset(lcms_dataset)
```

```
## Dataset summary:
## Valid dataset
## Description:  lc-ms mice spinal cord samples
## Type of data:  lcms-spectra
## Number of samples:  12
## Number of data points 410
## Number of metadata variables:  1
## Label of x-axis values:  mz/rt
## Label of data points:  intensity
## Number of missing values in data:  0
## Mean of data values:  1337915
## Median of data values:  279775
## Standard deviation:  3641583
## Range of values:  0 45173156
## Quantiles:
##       0%      25%      50%      75%     100%
##        0   111153   279775   845985 45173156
```

2. Data summary without statistics:

Hide

```
sum_dataset(lcms_dataset, stats=F)
```

```
## Dataset summary:
## Valid dataset
## Description:  lc-ms mice spinal cord samples
## Type of data:  lcms-spectra
## Number of samples:  12
## Number of data points 410
## Number of metadata variables:  1
## Label of x-axis values:  mz/rt
## Label of data points:  intensity
```

6.2 Boxplots

6.2.1 Boxplot of one or more variables distribution

6.2.1.1 Function to use

> boxplot_variables(dataset, variables = NULL, samples = NULL, horizontal = T, col = “lightblue”, nchar.label = 10, cex.axis = 0.8, ...)

- *dataset*: a specmine dataset;
- *variables*: vector with the names of the variables to plot. If no names are given (variables=NULL), all variables are plotted. Defaults to NULL;
- *samples*: vector with the names of the samples to take into consideration when plotting the variables. If no names are given all samples are taken into consideration (samples=NULL). Defaults to NULL;
- *horizontal*: boolean value (TRUE or FALSE) indicating if the boxplots should be drawn horizontaly or not. Defaults to TRUE;
- *col*: string that represents the color of the bodies of the boxplots. Can either be “lightblue” (default value),
- *nchar.label*: number of characters of variables’ names to display. If a variable has more than nchar.label characters, only the first nchar.label characters will be drawn. Defaults to 10;
- *cex.axis*: numeric value that indicates the amount by which the axis is magnified relative to the default. Defaults to 0.8;
- ...: additional parameters of boxplot function.

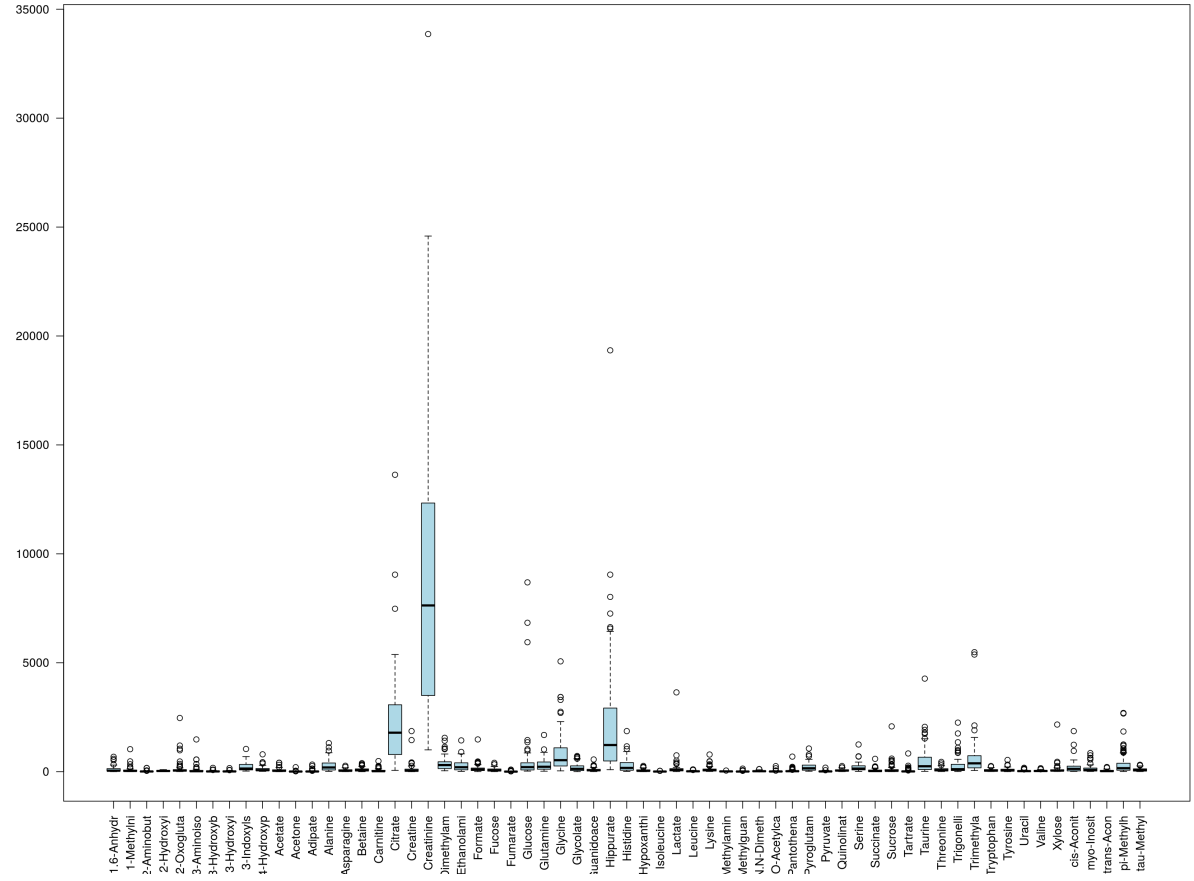
6.2.1.2 Example

Example makes use of an concentrations dataset (concentrations dataset used).

1. Boxplot of all variables distribution through all samples, with boxes diposed vertically

Hide

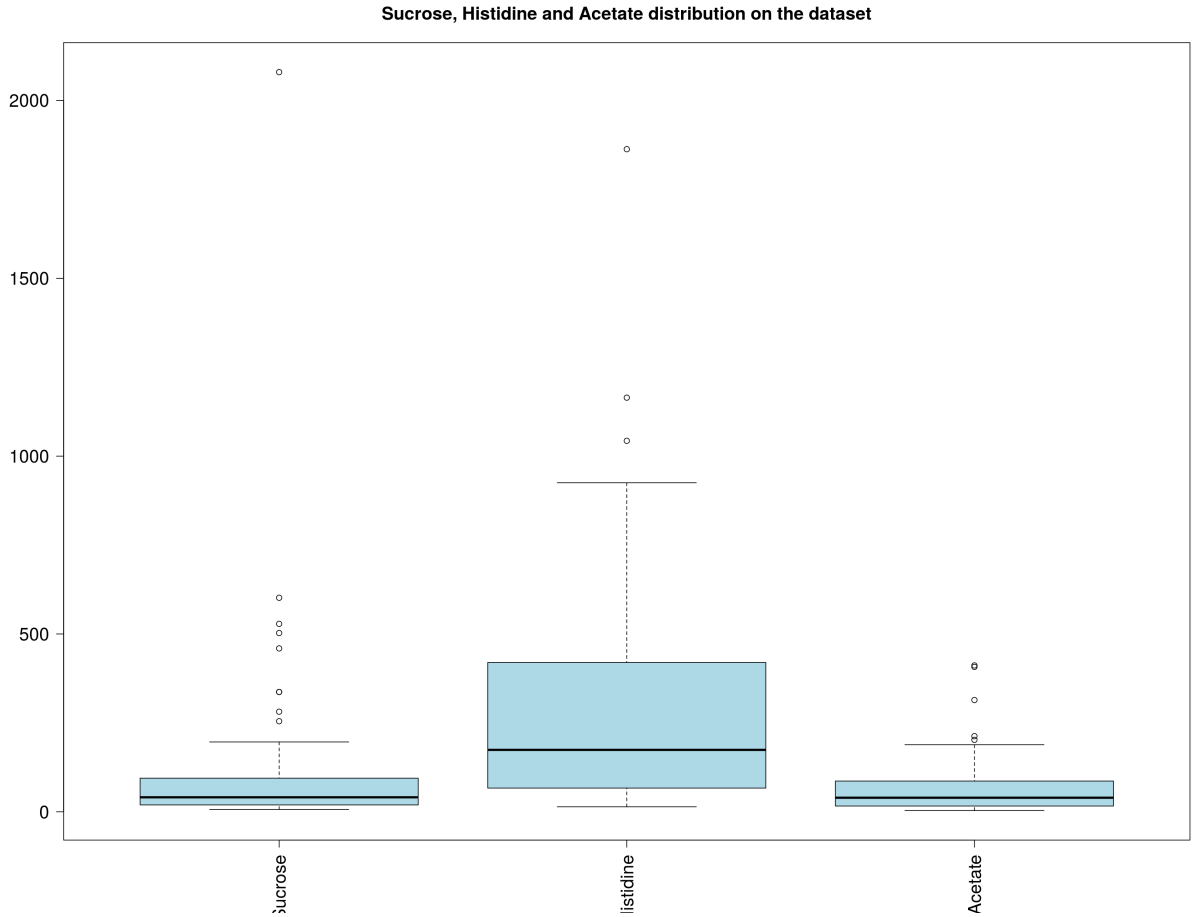
```
boxplot_variables(concentrations_dataset, horizontal=F, nhar.label=50, cex.axis=1)
```



2. Boxplot of “Sucrose”, “Histidine” and “Acetate” distribution through all samples, with a title

Hide

```
boxplot_variables(concentrations_dataset, variables=c("Sucrose", "Histidine", "Acetate"), horizontal=F, nhar.label=50, cex.axis=1.5, c
ex.main=1.5, main="Sucrose, Histidine and Acetate distribution on the dataset")
```



6.2.2 Boxplot of a variable distribution over two metadata variables

This function can only be used for datasets that have more than one metadata variable.

6.2.2.1 Function to use

> plotvar_twofactor(dataset, variable, meta.var1, meta.var2, colour = “darkblue”, title = “”, xlabel = NULL, ylabel = NULL)

- *dataset*: a specmine dataset;
- *variable*: name of the variable to plot;
- *meta.var1*: the name of one of the metadata variables to plot;
- *meta.var2*: the name of a different metadata variable to plot from the first one;
- *colour*: Colour of the boxes. Defaults to “darkblue”;
- *title*: title of the plot; **optional**
- *xlabel*: label of the xx axis; **optional**
- *ylabel*: label of the yy axis; **optional**

6.2.2.2 Example

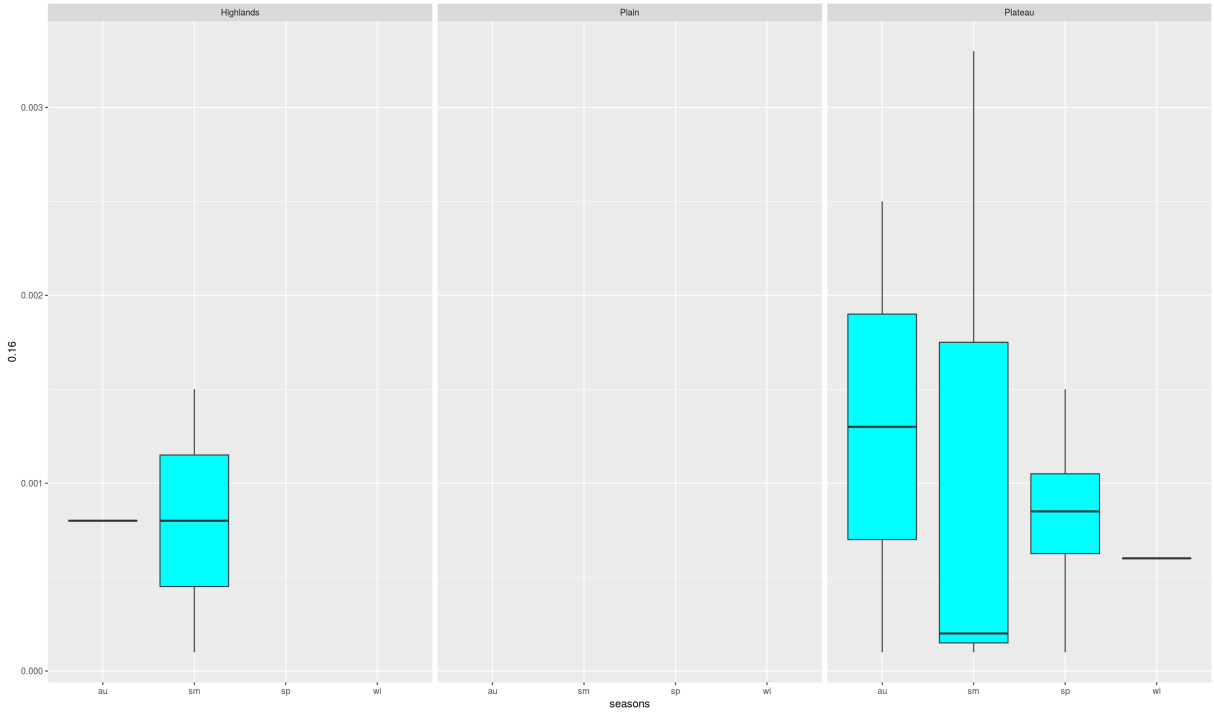
Example makes use of an nmr-peaks dataset (NMR peaks dataset used).

1. Plot the intensity distribution of the variable 0.16 ppm over the metadata variables “seasons” and “agroregions”

Hide

```
plotvar_twofactor(nmr_peaks_dataset, "0.16", "seasons", "agroregions", colour = "cyan")
```

```
## Warning: Removed 46 rows containing non-finite values (stat_boxplot).
```



In this case, as warned, some samples had to be excluded to produce the plot, as there were missing values.

6.2.3 Boxplot of one or more variables distribution over one metadata variable

6.2.3.1 Function to use

> boxplot_vars_factor(dataset, meta.var, variables = NULL, samples = NULL, horizontal = F, nchar.label = 10, col = NULL, vec.par = NULL, cex.axis = 0.8, ylabs = NULL, ...)

- *dataset*: a specmine dataset;
- *meta.var*: name of the metadata variable to plot;
- *variables*: names of the variables to plot;
- *samples*: vector with the names of the samples to take into consideration when plotting the variables. If no names are given all samples are taken into consideration (samples=NULL). Defaults to NULL;
- *horizontal*: boolean value (TRUE or FALSE) indicating if the boxplots should be drawn horizontaly or not. Defaults to TRUE;
- *nchar.label*: number of characters of variables' names to display. If a variable has more than nchar.label characters, only the first nchar.label characters will be drawn. Defaults to 10;
- *col*: vector with the different colors for each box (boxes of the same class of metadata variable will have the same color - *example 2.*). If not provided, default colors will be used, being in this case one different color for all boxes of a data variable (*example 1.*). Defaults to NULL;
- *vec.par*: if more than one variable is chosen, a vector specifying the disposition of the plots could be useful. This vector should have two elements - c(number_rows, number_columns);
- *cex.axis*: numeric value that indicates the amount by which the axis is magnified relative to the default. Defaults to 0.8;
- *ylabs*: yy axis label; **optional**
    - ...: additional parameters of boxplot function.

6.2.3.2 Example

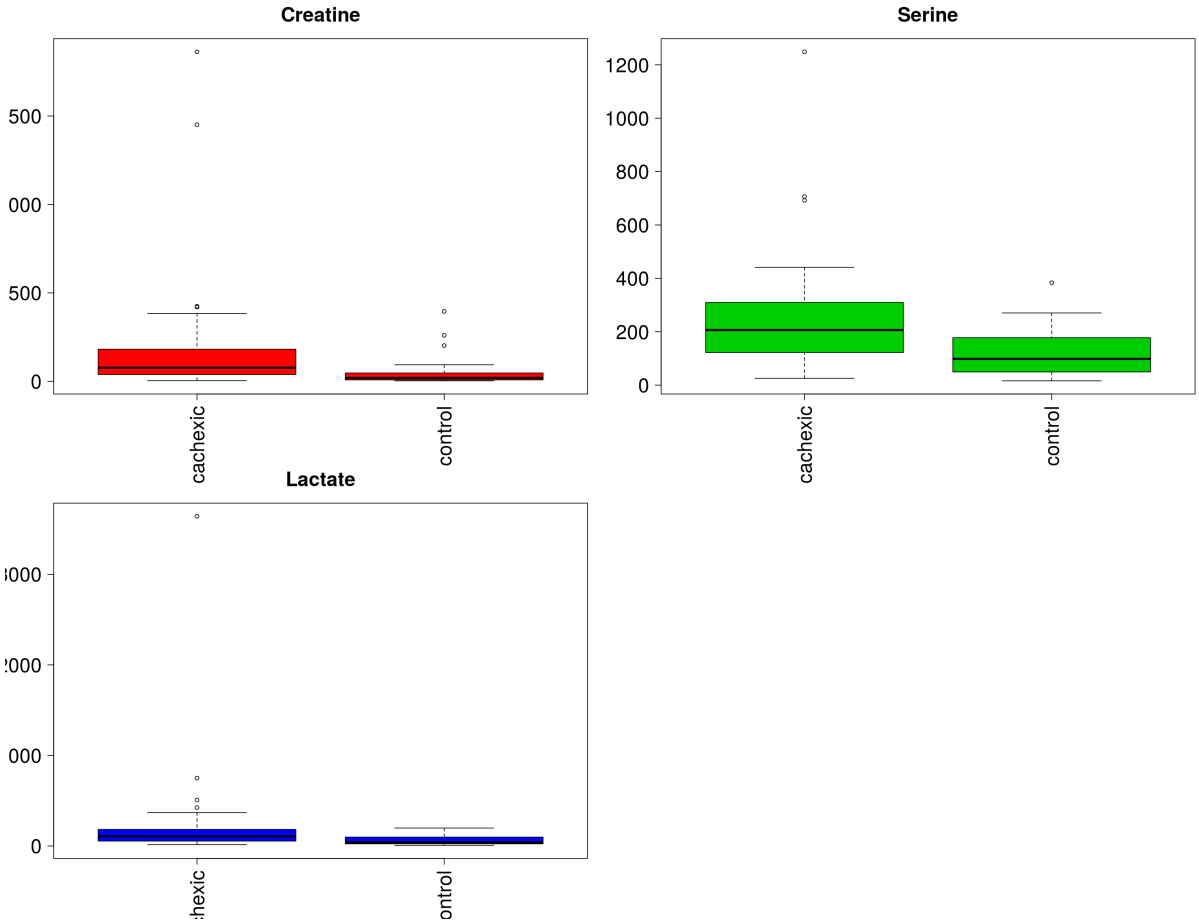
Example makes use of an concentrations dataset (concentrations dataset used).

1. Boxplot of three data variables (“Creatine”, “Serine” and “Lactate”) over one metadata variable (“Muscle.loss”). The plots of the two first variables will be plotted in

the first row, side by side, and the other one in the second row. Boxes here are colored according to the data variable they represent.

Hide

```
boxplot_vars_factor(concentrations_dataset, "Muscle.loss", variables = c("Creatine","Serine", "Lactate"), cex.axis=2, cex.main=2, vec.
par=c(2,2))
```

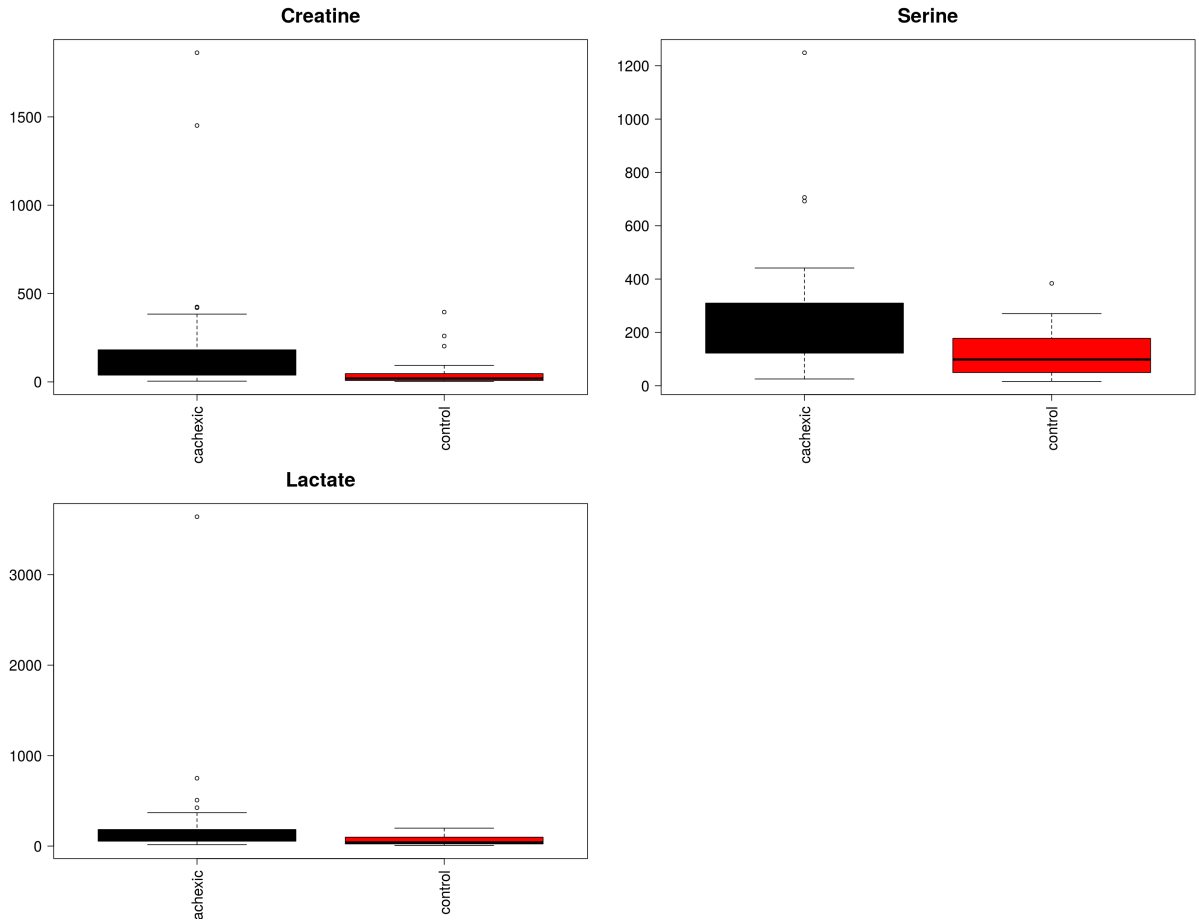


2. Boxplot of three data variables ("Creatine", "Serine" and "Lactate") over one metadata variable ("Muscle.loss"). The plots of the two first variables will be plotted in the first row, side by side, and the other one in the second row.Boxes here are colored according to the metadata class they represent.

Here, colors are set automaticaly:

Hide

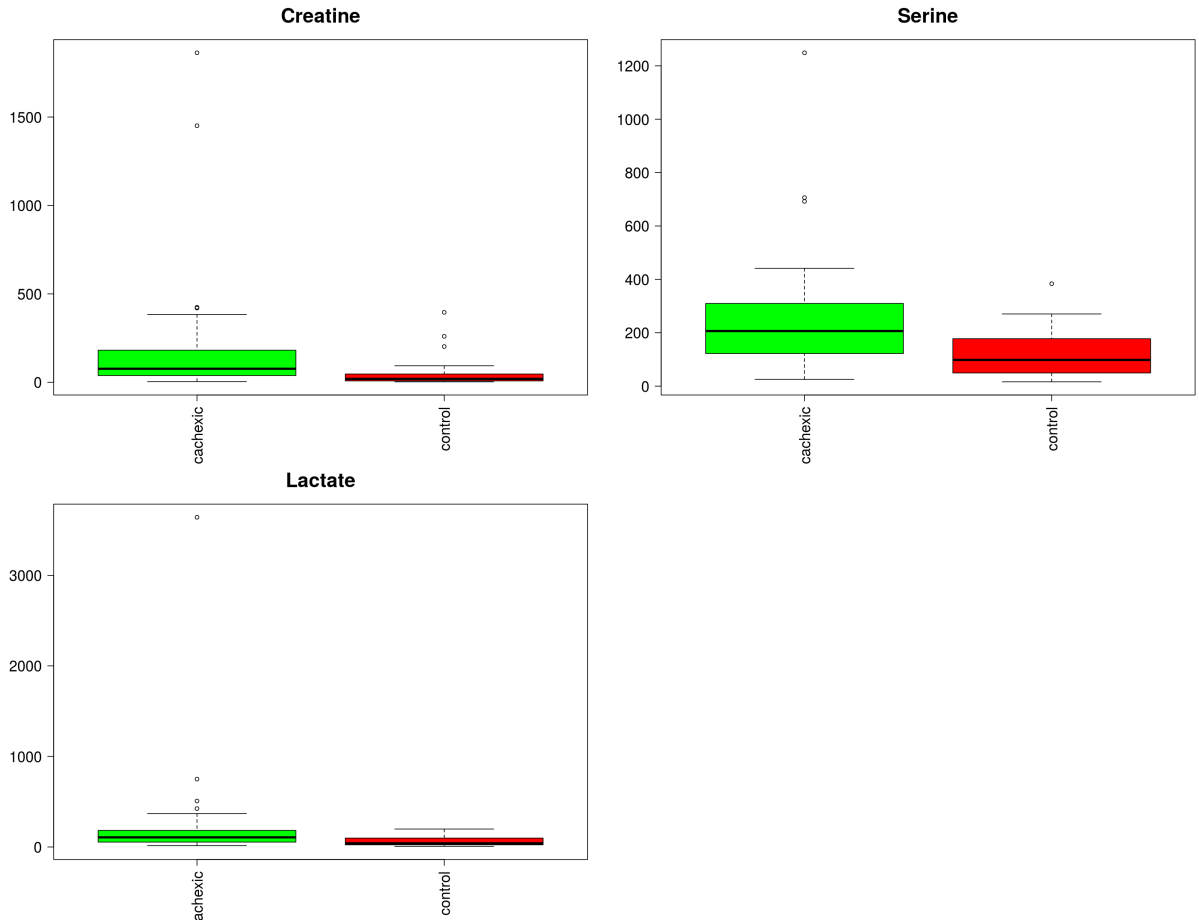
```
boxplot_vars_factor(concentrations_dataset, "Muscle.loss", variables = c("Creatine","Serine", "Lactate"), cex.axis=1.5, cex.main=2, ve
c.par=c(2,2), col=1:length(levels(get_metadata(concentrations_dataset)[,"Muscle.loss"])))
```



And here, two specific colors are chosen (as there are only two classes on the metadata variable chosen). If you don't know how many classes there are, you can type similarly to the following: length(levels(get_metadata(concentrations_dataset)[,"Muscle.loss"])).

Hide

```
boxplot_vars_factor(concentrations_dataset, "Muscle.loss", variables = c("Creatine","Serine", "Lactate"), cex.axis=1.5, cex.main=2, ve
c.par=c(2,2), col=c("green", "red"))
```



6.3 Spectra plot

Only makes sense for data of spectral type.

6.3.1 Function to use

> plot_spectra(dataset, column.class, func = NULL, samples = NULL, variable.bounds = NULL, xlab = NULL, ylab = NULL, lty = 1, legend.place = "topright", cex = 0.8, reverse.x = F, ...)

- *dataset*: a specmine dataset;
- *meta.var*: name of a metadata variable by which each spectrum sample will be colored;
- *fun*: function to compute the summary statistics to apply to the data. Defaults to NULL; **optional**
- *samples*: vector with the names of the samples to plot. If no names are given all samples are plotted (samples=NULL). Defaults to NULL;
- *variable.bounds*: numeric vector with two elements indicating the interval of x-values to plot; **optional**
- *xlab*: xx axis label; **optional**
- *ylab*: yy axis label; **optional**
- *legend.place*: string indicating the place where the legend's box will be placed. Can either be: "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or "center";
- *cex*: numeric value indicating the relative size of the legend. Defaults to 0.8;
- *reverse.x*: boolean value (TRUE or FALSE) ndicating if the x-axis will be shown reversed (decreasingly) or not. Defaults to FALSE;
- *lty*: line type (parameter of matplot function);
- ...: additional parameters of matplot function.

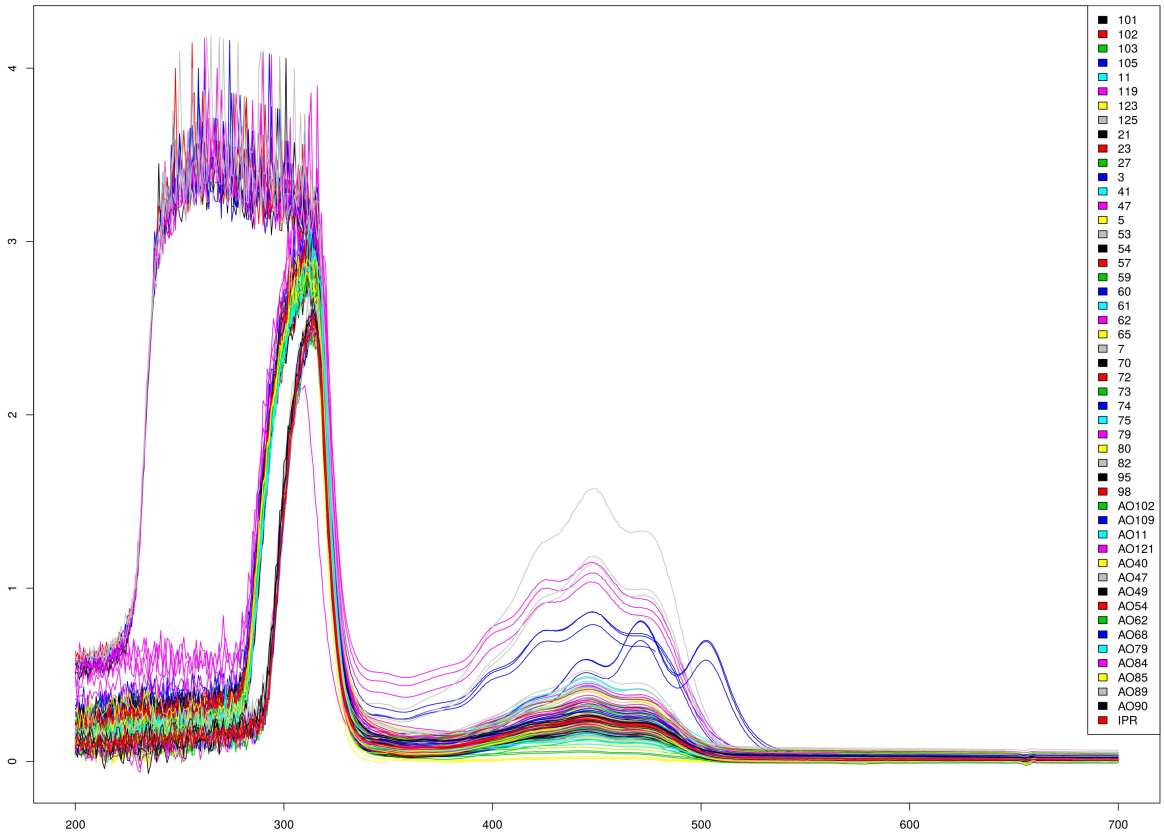
6.3.2 Example

Example makes use of an UV-Vis dataset (UV-Vis dataset used).

1: Spectra plot of all samples, coloured by a metadata variable (here is "varieties"):

Hide

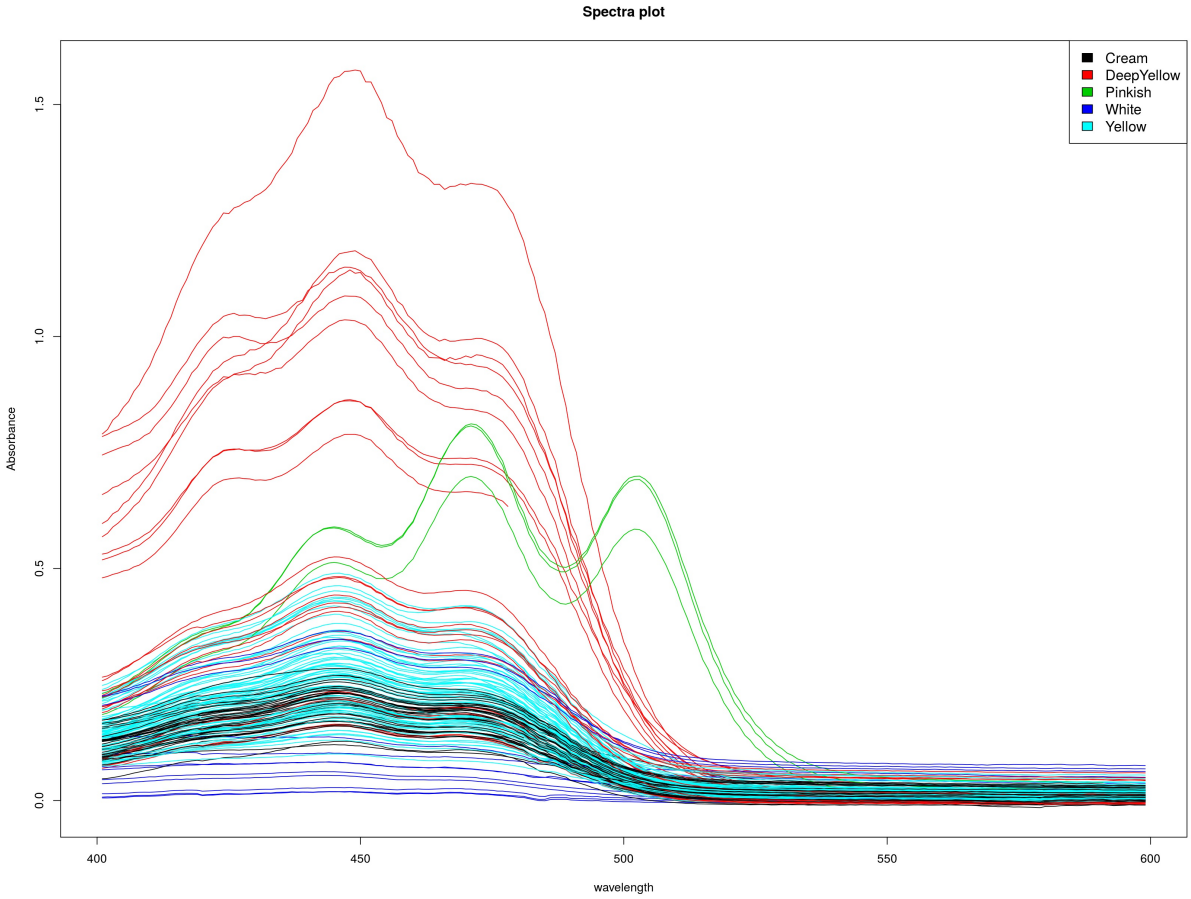
```
plot_spectra(uvvis_spectra_dataset, "varieties", cex=1)
```



2: Spectra plot of all samples, coloured by a metadata variable (here is "colors"), between the xx values 400 and 600, giving labels to the axis and a title:

Hide

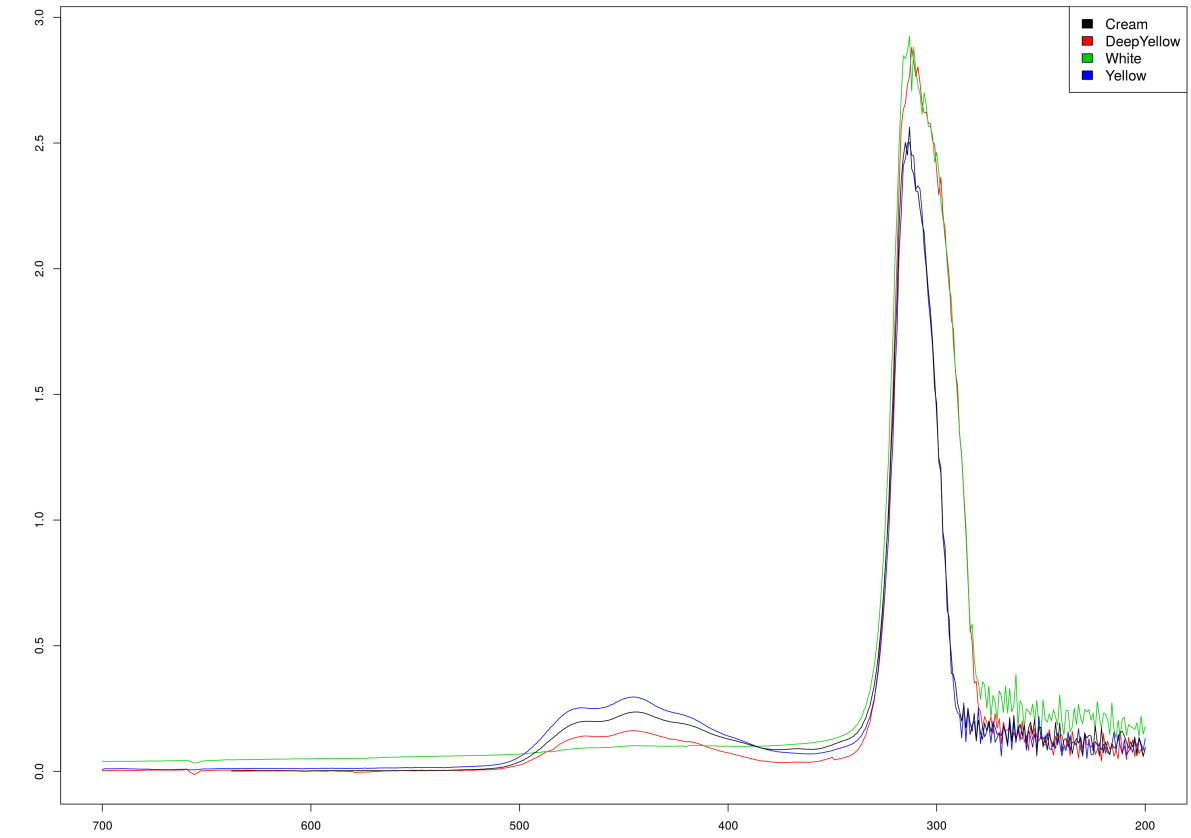
```
plot_spectra(uvvis_spectra_dataset, "colors", variable.bounds=c(400,600), xlab="wavelength", ylab="Absorbance", main="Spectra plot", cex=1.2)
```



3: Spectra plot of four samples (one from each color), coloured by the metadata variable "colors", with the xx axis values decreasing:

Hide

```
plot_spectra(uvvis_spectra_dataset, "colors", samples=c("3.1", "5.1", "7.1", "11.1"), reverse.x=T, cex=1.2)
```



6.4 Peaks Plot

6.4.1 Function to use

> plot_peaks(dataset, column.class, func = NULL, samples = NULL, variable.bounds = NULL, xlab = NULL, ylab = NULL, legend.place = "topright", cex = 0.8, reverse.x = F, p.size=0.5, ...)

- *dataset*: a specmine dataset;
- *meta.var*: name of a metadata variable by which each spectrum sample will be colored;
- *samples*: vector with the names of the samples to plot. If no names are given all samples are plotted (samples=NULL). Defaults to NULL;
- *variable.bounds*: numeric vector with two elements indicating the interval of x-values to plot; **optional**
- *xlab*: xx axis label; **optional**
- *ylab*: yy axis label; **optional**
- *legend.place*: string indicating the place where the legend's box will be placed. Can either be: "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or "center";
- *cex*: numeric value indicating the relative size of the legend. Defaults to 0.8;
- *reverse.x*: boolean value (TRUE or FALSE) ndicating if the x-axis will be shown reversed (decreasingly) or not. Defaults to FALSE;
- *p.size*: numeric value indicating the relative size of the plot points. Defaults to 0.5;
- ...: additional parameters of matplot function.

6.5 Samples' Statistics

6.5.1 Function to use

> stats_by_sample(dataset, samples = NULL)

- *dataset*: s specmine dataset;
- *samples*: vector with the names of the samples to take into consideration when calculating the statistics. If no names are given all samples are used (samples=NULL). Defaults to NULL.

6.5.2 Example

Example makes use of an concentrations dataset (concentrations dataset used).

1: Statistics on all samples:

Hide

```
samps.stats=stats_by_sample(concentrations_dataset)
DT::datatable(samps.stats, options=list(scrollX=TRUE))
```

6.6 Variable's Statistics

6.6.1 Function to use

> stats_by_variable(dataset, variables = NULL, variable.bounds = NULL)

- *dataset*: a specmine dataset;
- *variables*: vector with the variables to use to calculate the statistics. If numeric vector, it is assumed that they represent indexes. If no variables are given, all are used (variables=NULL). Defaults to NULL;
- *variable.bounds*: if the variables are numeric, you can choose to select the interval of variables to take into consideration, so that you do not need to select every single data point wanted. Again, if no variables are given, all are used (variables=NULL). Defaults to NULL.

6.6.2 Example

Example makes use of an concentrations dataset (concentrations dataset used).

1: Statistics on all variables:

Hide

```
vars.stats=stats_by_variable(concentrations_dataset)
DT::datatable(vars.stats, options=list(scrollX=TRUE))
```

7 Pre-Processing

7.1 Missing Values

The treatment of missing values can be done by replacing them with another value.

> missingvalues_imputation(dataset, method = "value", value = 5e-04, k = 5)

- *dataset*: a specmine dataset;
- *method*: method by which to treat the missing values. It can either be
  - "value": replaces the missing values with a specific value (given in *value* argument);
  - "mean": replaces the missing values with the mean of the variables' values;
  - "median": replaces the missing values with the median of the variables' values
  - "knn": replaces the missing values with k nearest neighbor (given in *k* argument) averaging;
  - "linapprox": replaces the missing values with linear approximation.

- *value*: the value to replace the missing values if the method "value" is chosen;
- *k*: the number of neighbors if the method "knn" is chosen.

7.2 Remove Data

7.2.1 Remove data variables

7.2.1.1 Function to use

You can remove specific data variables from the dataset

> remove_data_variables(dataset, variables.to.remove, by.index = FALSE)

- *dataset*: a specmine dataset;
- *variables.to.remove*: vector with the indexes or names of the variables to remove;
- *by.index*: boolean value (TRUE or FALSE) indicating if the values in *variables.to.remove* argument are indexes or not. Defaults to FALSE;

You can remove an interval of data variables. Should not be used for concentrations data

> remove_x_values_by_interval(dataset, min.value, max.value)

- *dataset*: a specmine dataset;
- *min.value*: the minimum value of the interval;
- *max.value*: the maximum value of the interval.

7.2.1.2 Example

Example makes use of a concentrations dataset (concentrations dataset used).

1. Remove the variables *Creatine* and *Serine* from the dataset:

Hide

```
remove_vars_concentrations_dataset = remove_data_variables(concentrations_dataset, c("Creatine","Serine"))
```

2. Data table after removing the variables from the dataset:

Hide

```
DT::datatable(remove_vars_concentrations_dataset$data, options=list(scrollX = TRUE))
```

7.2.2 Remove metadata variables

7.2.2.1 Function to use

> remove_metadata_variables(dataset, variables.to.remove)

- *dataset*: a specmine dataset;
- *variables.to.remove*: vector with the names of the metadata variables to remove.

7.2.2.2 Example

Example makes use of an IR dataset (IR dataset used).

1. Remove the metadata variable *varieties* from the dataset:

Hide

```
remove_meta_ir_spectra_dataset=remove_metadata_variables(ir_spectra_dataset, "varieties")
```

2. Metadata table after removing the metadata variables from the dataset:

Hide

```
DT::datatable(remove_meta_ir_spectra_dataset$metadata, options=list(scrollX = TRUE))
```

7.2.3 Remove samples

7.2.3.1 Function to use

> remove_samples(dataset, samples.to.remove, rebuild.factors = T)

- *dataset*: a specmine dataset;
- *samples.to.remove*: vector with the names of the samples to remove from the dataset;
- *rebuild.factors*: boolean value (TRUE or FALSE) indicating if the metadata factors should be rebuilded after the subset. Defaults to TRUE.

7.2.3.2 Example

Example makes use of a concentrations dataset (concentrations dataset used).

1. Remove the variables *PIF_178* and *PIF_090* from the dataset:

Hide

```
remove_samps_concentrations_dataset = remove_samples(concentrations_dataset, c("PIF_178","PIF_090"))
```

2. Data table after removing the samples from the dataset:

Hide

```
DT::datatable(remove_samps_concentrations_dataset$data, options=list(scrollX = TRUE))
```

Hide

```
DT::datatable(remove_samps_concentrations_dataset$metadata, options=list(scrollX = TRUE))
```

7.3 Remove Data according to presence of missing values

7.3.1 Remove samples

7.3.1.1 Functions to use

You can remove samples according to the amount of missing values present in each sample

> remove_samples_by_nas(dataset, max.nas = 0, by.percent = F)

- *dataset*: a specmine dataset;
- *max.nas*: maximum number or percentage of missing values that a sample can have. Samples with more missing values than the ones allowed are removed. Defaults to 0;
- *by.percent*: boolean value (TRUE or FALSE) indicating if the value of the *max.nas* argument is a percentage or not. Defaults to FALSE.

You can remove samples if the corresponding value of a metadata variable is missing

> remove_samples_by_na_metadata(dataset, metadata.var)

- *dataset*: a specmine dataset;
- *metadata.var*: name of the metadata variable by which the samples could be removed if the value is missing.

7.3.2 Remove variables

7.3.2.1 Function to use

You can remove variables according to the amount of missing values present in each variable

> remove_variables_by_nas(dataset, max.nas = 0, by.percent = F)

- *dataset*: a specmine dataset;
- *max.nas*: maximum number or percentage of missing values that a variable can have. Variables with more missing values than the ones allowed are removed. Defaults to 0;
- *by.percent*: boolean value (TRUE or FALSE) indicating if the value of the *max.nas* argument is a percentage or not. Defaults to FALSE.

7.4 Data Transformation

The data can be transformed by one of two methods: logarithmic transformation or cubic root transformation.

> transform_data(dataset, method = “log”)

- *dataset*: a specmine dataset;
- *method*: string specifying the transformation method. It can either be
  - “log”: logarithmic transformation;
  - “cubicroot”: cubic root transformation.

7.5 Scaling

The data can be scaled according to one of three methods: auto, range or parto scaling.

> scaling(dataset, method = “auto”)

- *dataset*: a specmine dataset;
- *method*: string specifying the scaling method. It can either be “auto”, “range” or “pareto”.

7.6 Correction

Three methods are available to perform data correction: background, offset and baseline corrections. This should only be applied to spectral data.

> data_correction(dataset, type = “background”, method = “modpolyfit”, ...)

- *dataset*: a specmine dataset;
- *type*: string specifying the type of correction that will be applied. It can either be “background”, “offset”, “baseline”;
- *method*: when “baseline” correction is chosen, you can choose the baseline method to use. It can either be “als”, “fillPeaks”, “irls”, “lowpass”, “medianWindow”, “modpolyfit”, “peakDetection”, “rfbaseline”, “rollingBal”, “shirley”. Defaults to “modpolyfit”;
- ...: Additional parameters that may be set, according to the method chosen. See the links provided in each method above for further information, as these are methods developed by the baseline package.

7.7 Smoothing Interpolation

Smoothing interpolation with methods Bin, Loess and Savitzky-Golay.

> smoothing_interpolation(dataset, method = “bin”, reducing.factor = 2, x.axis = NULL, p.order = 3, window = 11, deriv = 0)

- *dataset*: a specmine dataset;
- *method*: string specifying the smoothing method. It can either be “bin”, “loess”, “savitzky.golay”. It defaults to “bin”;
- *reducing.factor*: if method “bin” is chosen, the numeric value indicating the reducing factor should be given. It defaults to 2;
- *x.axis*: if method “loess” is chosen, a numeric vector representing the new x-axis for loess method can be given. Defaults to NULL;

- *p.order*: if method “savitzky.golay” is chosen, a numeric value representing the polynomial order should be given. It defaults to 3;
- *window*: if method “savitzky.golay” is chosen, a odd numeric value indicating the size of the window. It defaults to 11;
- *deriv*: if method “savitzky.golay” is chosen, a numeric value indicating the differentiation order. It defaults to 0.

7.8 Convert to Factor

Metadata variables can be converted to factors (useful in some analysis, for example, train classifcation models and color plots based on a metadata variable):

```
convert_to_factor(dataset, metadata.var)
```

- *dataset*: a specmine dataset;
- *metadata.var*: name of the metadata variable to convert.

7.9 Mean Centering

```
mean_centering(dataset)
```

- *dataset*: a sepcmine dataset.

7.10 First Derivative

```
first_derivative(dataset)
```

- *dataset*: a specmine dataset.

7.11 Multiplicative Scatter Correction

```
msc_correction(dataset)
```

- *dataset*: a specmine dataset.

7.12 Data Normalization

Data normalization can be performed through the sum, median, a reference sample or a reference variable

```
normalize(dataset, method, ref = NULL, constant = 1000)
```

- *dataset*: a specmine dataset;
- _ method_: string specifying the normalization method. It can either be
  - “sum”: normalization by the sum of a constant, given in the *constant* argument;
  - “median”: normalization by the median;
  - “ref.sample”: normalization by a reference sample, given in the *ref* argument;
  - “ref.feature”: normalization by a reference variable, given in the *ref* argument.
- *ref*: if method “ref.sample” or ref.feature“” is chosen, a string indicating the sample or variable of reference must be given;
- *constant*: if method “sum” is chosen, the constant by which to do the sum normalization must be given.

7.13 Subset dataset by variables

7.13.1 Functions to use

You can subset the dataset by only keeping data from specific variables

```
subset_x_values(dataset, variables, by.index = FALSE)
```

- *dataset*: a specmine dataset;
- *variables*: vector with the indexes or names of the variables to keep;
- *by.index*: boolean value (TRUE or FALSE) indicating if the values in *variables* argument are indexes or not. Defaults to FALSE;

You can subset the dataset by keeping data from an interval of variables. Only for data whose variables are numeric, i.e., should not be used for concentrations data

```
subset_x_values_by_interval(dataset, min.value, max.value)
```

- *dataset*: a specmine dataset;
- *min.value*: the minimum value of the interval;
- *max.value*: the maximum value of the interval.

7.13.2 Examples

Example makes use of an UV-Vis dataset (UV-Vis dataset used).

1. Subset dataset to only keep the wavelength values between 400 and 600:

Hide

```
subset_vals_int_uvvis_spectra_dataset=subset_x_values_by_interval(uvvis_spectra_dataset, 400, 600)
```

2. Data and metadata tables after subsetting the dataset:

Hide

```
DT::datatable(subset_vals_int_uvvis_spectra_dataset$data, options=list(scrollX = TRUE))
```

Hide

```
DT::datatable(subset_vals_int_uvvis_spectra_dataset$metadata, options=list(scrollX = TRUE))
```

7.14 Subset dataset by samples

7.14.1 Functions to use

You can subset the dataset by only keeping data from specific samples

> subset_samples(dataset, samples, rebuild.factors = T)

- *dataset*: a specmine datatset;
- *samples*: vector with the indexes or names of the samples to keep;
- *rebuild.factors*: boolean value (TRUE or FALSE) indicating if the metadata factors should be rebuilded after the subset. Defaults to TRUE.

You can subset the dataset by keeping samples that have certain metadata classe(s)

> subset_samples_by_metadata_values(dataset, metadata.varname, values)

- *dataset*: a specmine dataset;
- *metadata.varname*: string representing the metadata variable by whose values you will want to keep the samples;
- *values*: the samples that are kept must have the metadata classe(s) of the metadata variable chosen (metadata.varname) specified in this argument, using a character vector with the classe(s).

7.14.2 Examples

Example makes use of an nmr-peaks dataset (NMR peaks dataset used).

1. Subset dataset to only keep the samples *AC_au*, *AC_sm*, *AC_sp*, *AC_wi*:

Hide

```
subset_samp_nmr_peaks_dataset=subset_samples(nmr_peaks_dataset, c("AC_au", "AC_sm", "AC_sp", "AC_wi"))
```

2. Data and metadata tables after subsetting the dataset:

Hide

```
DT::datatable(subset_samp_nmr_peaks_dataset$data, options=list(scrollX = TRUE))
```

Hide

```
DT::datatable(subset_samp_nmr_peaks_dataset$metadata, options=list(scrollX = TRUE))
```

3. Subset dataset to only keep samples from winter (*wi*) and summer (*sm*) seasons:

Hide

```
subset_samp_met_nmr_peaks_dataset=subset_samples_by_metadata_values(nmr_peaks_dataset, "seasons", c("wi", "sm"))
```

4. Data and metadata tables after subsetting the dataset:

Hide

```
DT::datatable(subset_samp_met_nmr_peaks_dataset$data, options=list(scrollX = TRUE))
```

Hide

```
DT::datatable(subset_samp_met_nmr_peaks_dataset$metadata, options=list(scrollX = TRUE))
```

7.15 Subset dataset by samples and variables

7.15.1 Functions to use

You can subset the datast by keeping specific samples and variables

> subset_by_samples_and_xvalues(dataset, samples, variables = NULL, by.index = F, variable.bounds = NULL, rebuild.factors = T)

- *dataset*: a specmine dataset;
- *samples*: vector with the indexes or names of the samples to keep;
- *variables*: vector with the indexes or names of the variables to keep. Defaults to NULL;
- *by.index*: boolean value (TRUE or FALSE) indicating if the values in *variables* and *samples* arguments are indexes or not. Defaults to FALSE;
- *variable.bounds*: numeric vector of two elements with the minimum and maximum values of the interval of variables to keep. Argument should not be used for concentrations dataset. When used, arguments *by.index* should be FALSE and *variables* NULL. Defaults to NULL;
- *rebuild.factors*: boolean value (TRUE or FALSE) indicating if metadata factors should be rebuilded. Defaults to TRUE.

7.15.2 Examples

Example makes use of an concentrations dataset (concentrations dataset used).

1. Subset the dataset to only keep the data from samples *PIF_178* and *NETL_022_V1*, and from variables *Creatinine* and *Serine*.

Hide

```
subset_vars_samp_concentrations_dataset = subset_by_samples_and_xvalues(concentrations_dataset, c("PIF_178","NETL_022_V1"), variables = c("Creatine","Serine"))
```

2. Data and metadata tables after subsetting the dataset:

Hide

```
DT::datatable(subset_vars_samp_concentrations_dataset$data, options=list(scrollX = TRUE))
```

Hide

```
DT::datatable(subset_vars_samp_concentrations_dataset$metadata, options=list(scrollX = TRUE))
```

7.16 Low-level Data Fusion

7.16.1 Function to use

Datasets from different types of data can be joined together. Only the samples that are the same in both datasets will appear in the final one.

> low_level_fusion(datasets)

- *datasets*: List of specmine datasets to join in one.

7.17 Aggregate Samples

7.17.1 Function to use

> aggregate_samples(dataset, indexes, aggreg.fn = "mean", meta.to.remove = c())

- *dataset*: a specmine dataset;
- *indexes*: numeric vector indicating how samples will be grouped. For example, the vector c(1,2,2,3,1,3) indicates that the first and fifth samples are grouped in one, the second and third in another one, and the forth and last one are aggregated in another one;
- *aggreg.fn*: string representing the name of the function by which the samples will be aggregated. It can either be "mean", "median", "sum", "max" (maximum value) or "min" (minimum value). It defaults to "mean";
- *meta.to.remove*: vector with the name(s) of the variable(s) you want to remove after the aggrgation. **optional**

7.17.2 Examples

Example makes use of an nmr-peaks dataset (NMR peaks dataset used).

1. In the following example, the samples are aggregated according to the classes in the metadata variable "seasons", i.e., samples with the same class on the metadata variable will be grouped together.

Hide

```
#Number of different metadata variable's values will be the number of different groups in which the samples will be grouped:
x=1:length(levels(nmr_peaks_dataset$metadata[["seasons"]]))
names(x)=levels(nmr_peaks_dataset$metadata[["seasons"]])
indexes=c()
for (meta in nmr_peaks_dataset$metadata[["seasons"]]){
  indexes=c(indexes, x[meta])
}

#The "agroregions" metadata variable is removed after samples are grouped, as it stops making sense for it to be there
aggregated_nmr_peaks_dataset = aggregate_samples(nmr_peaks_dataset, indexes, meta.to.remove="agroregions")
```

2. Data and Metadata tables after aggregation:

Hide

```
DT::datatable(aggregated_nmr_peaks_dataset$data, options=list(scrollX=TRUE))
```

Hide

```
DT::datatable(aggregated_nmr_peaks_dataset$metadata, options=list(scrollX=TRUE))
```

7.18 Flat Pattern Filter

7.18.1 Function to use

> flat_pattern_filter(dataset, filter.function = "iqr", by.percent = T, by.threshold = F, red.value = 0)

- *dataset*: a specmine dataset;
- *filter.function*: function by which to filter the data variables. It can either be
  - "iqr": Interquantile Range;
  - "rsd": Relative Standard Deviation;
  - "mad": Median Absolute Deviation;
  - "mean": Mean;
  - "median": Median.

- *by.percent*: boolean value (TRUE or FALSE) indicating if the amount of variables to filter in the dataset is given in a percentage. It defaults to TRUE;
- *by.threshold*: boolean value (TRUE or FALSE) indicating if the amount of variables to filter in the dataset is defined by wether they are under or not a given threshold. It defaults to FALSE;
- *red.value*: numeric value indicating a percentage or threshold. It can also be "auto", indicating that the number of variables to remove are calculated automatically.

7.19 Replace specific data or metadata values in the dataset

7.19.1 Functions to use

You can replace a data value for a new one on the dataset

```
replace_data_value(dataset, x.axis.val, sample, new.value, by.index = F)
```

- *dataset*: a specmine dataset;
- *x.axis.val*: index or name of the variable that corresponds to the data point to replace;
- *sample*: name of the sample that corresponds to the data point to replace;
- *new.value*: new value (numeric) of the data point;
- *by.index*: boolean value (TRUE or FALSE) indicating if the value in *x.axis.val* argument is an index or not. Defaults to FALSE;

You can replace a metadata value for a new one on the dataset

```
replace_metadata_value(dataset, variable, sample, new.value)
```

- *dataset*: a specmine dataset;
- *variable*: name of the metadata variable that corresponds to the metadata point to replace;
- *sample*: name of the sample that corresponds to the metadata point to replace;
- *new.value*: new value (numeric or character string) of the metadata point.

8 Univariate Analysis

8.1 T-test

8.1.1 Function to use

```
tTests_dataset(dataset, metadata.var, threshold = NULL, write.file = F, file.out = "ttests.csv")
```

- *dataset*: a specmine dataset;
- *metadata.var*: metadata variable to use in the t-tests. It is tested the differences in means of the variables between the samples that belong to the different groups of the metadata variable given;
- *threshold*: threshold value of the p-value. Only the results of the variables whose p-values are under the threshold are returned; **optional**
- *write.file*: boolean value (TRUE or FALSE) indicating if you want the results to be written in a file. Defaults to FALSE;
- *file.out*: string with the file path to write to. You only need to give this information if you chosen to write the results to a file (write.file=T). Defaults to "ttests.csv".

```
plot_ttests(dataset, tt.results, tt.threshold = 0.01)
```

- *dataset*: the specmine dataset that led to the results of the t-tests given in *tt.results* argument;
- *tt.results*: variable containing the t-tests results, obatined from tTests_dataset function;
- *tt.treshold*: numeric value indicating the p-value treshold. An horizontal line will be drawn in the plot and the data points whose p-values are under the treshold will be blue, while the other ones will be grey. Defaults to 0.01.

8.1.2 Examples

Example makes use of an concentrations dataset (concentrations dataset used).

1. Perform t-tests on the variables of the dataset, testing if they are significantly different between the classes of the metadata variable *Muscle.loss*

Hide

```
res_ttest_concentrations = tTests_dataset(concentrations_dataset, "Muscle.loss")
```

2. Table with the results obtained:

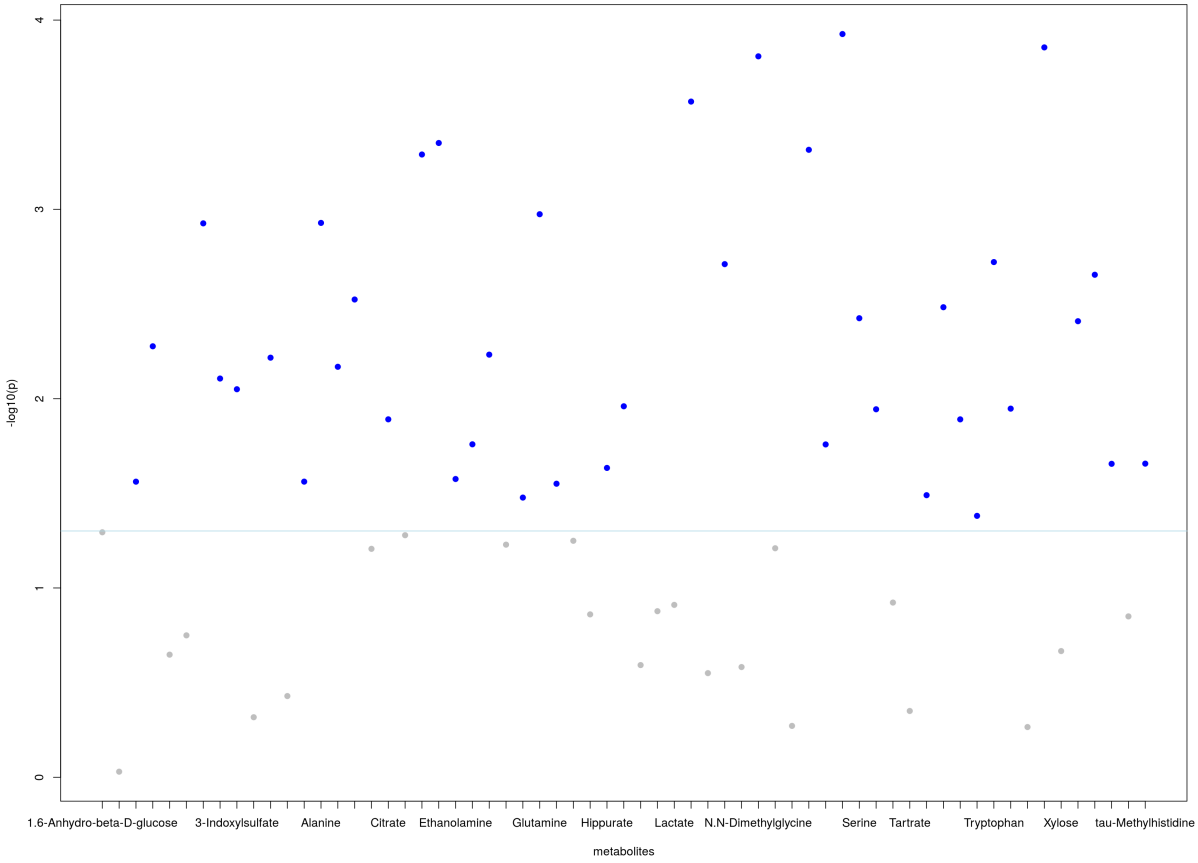
Hide

```
DT::datatable(res_ttest_concentrations)
```

3. Plot of the results, with a p-value treshold of 0.05:

Hide

```
plot_ttests(concentrations_dataset, res_ttest_concentrations, tt.threshold = 0.05)
```



8.2 One-Way ANOVA

8.2.1 Function to use

> aov_all_vars(dataset, column.class, doTukey = T, write.file = F, file.out = "anova-res.csv")

- *dataset*: a specmine dataset;
- *column.class*: metadata variable to use in the ANOVA. It is tested the differences in means of the variables between the samples that belong to the different groups of the metadata variable given;
- *doTukey*: boolean value (TRUE or FALSE) indicating if TukeyHSD test should be performed or not. Defaults to TRUE;
- *write.file*: boolean value (TRUE or FALSE) indicating if you want the results to be written in a file. Defauls to FALSE;
- *file.out*: string with the file path to write to. You only need to give this information if you chosen to write the results to a file (write.file=T). Defaults to "anova-res.csv".

> plot_anova(dataset, anova.results, anova.threshold = 0.01, reverse.x = F)

- *dataset*: the specmine dataset that led to the results of the t-tests given in *anova.results* argument;
- *anova.results*: variable containing the one-way ANOVA results, obatined from aov_all_vars function;
- *anova.treshold*: numeric value indicating he p-value treshold. An horizontal line will be drawn in the plot and the data points whose p-values are under the treshold will be blue, while the other ones will be grey. Defaults to 0.01;
- *reverse.x*: boolean value (TRUE or FALSE) indicating if the x-axis should be plotted in reverse or not. Defaults to FALSE.

8.2.2 Examples

Example makes use of an IR dataset (IR dataset used).

1. To perform this test, the original dataset was filtered, using interquantile range function and filtering 75% of the variables:

Hide

```
ir_spectra_dataset_fpf=flat_pattern_filter(ir_spectra_dataset, red.value=75)
```

2. Perform one-way ANOVA tests on the variables of the dataset, testing if they are significantly different between the classes of the metadata variable *varieties*, without performing the TukeyHSD test:

Hide

```
res_one_anova_ir = aov_all_vars(ir_spectra_dataset_fpf, "varieties", doTukey = FALSE)
```

3. Table with the results obtained:

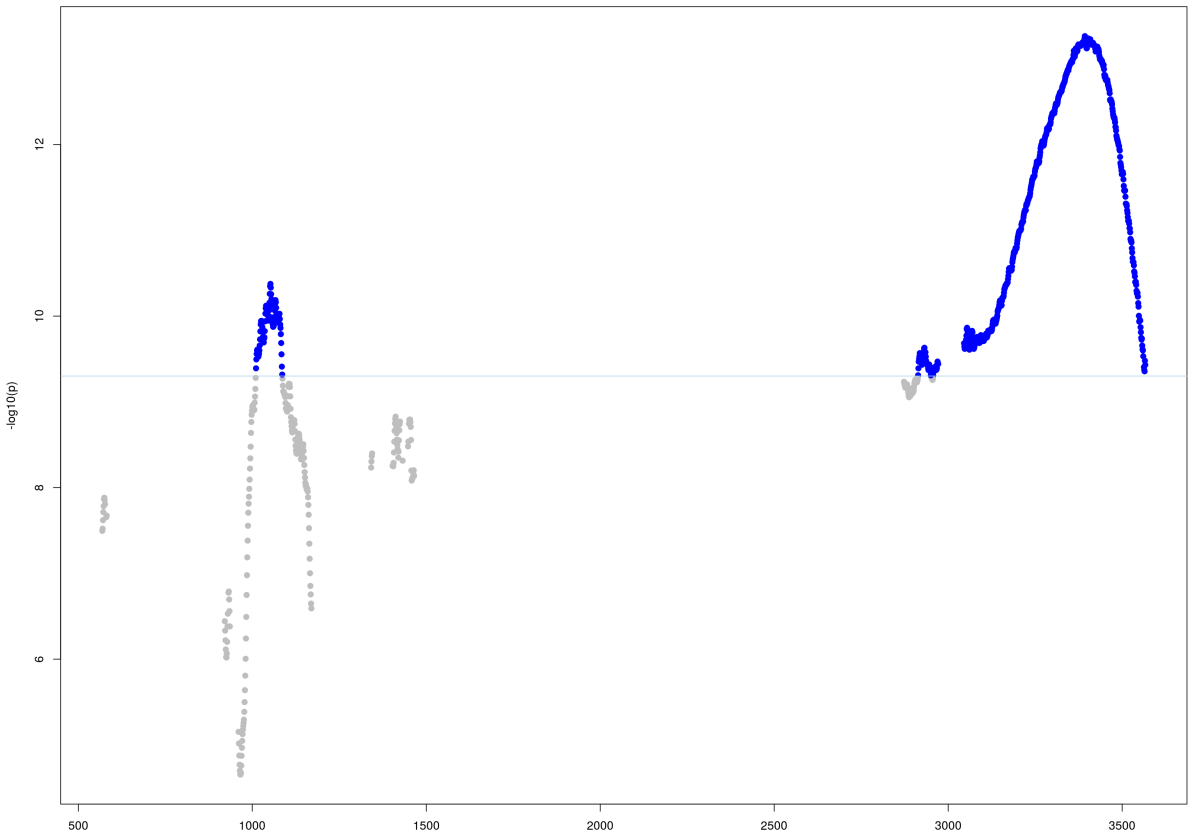
Hide

```
DT::datatable(res_one_anova_ir)
```

4. Plot of the results, with a treshold of 0.05*10^-8:

Hide

```
plot_anova(ir_spectra_dataset_fpf, res_one_anova_ir, 0.05e-8)
```



8.3 Multifactor ANOVA

8.3.1 Functions to use

To perform multifactor ANOVA:

> multifactor_aov_all_vars(dataset, metadata.vars, combination)

- *dataset*: a specmine dataset;
- *metadata.vars*: character vector with strings representing the metadata variables to use in multifactor ANOVA. It is tested the differences in means of the variables between the samples that belong to the different groups of the metadata variables given;
- *combination*: string representating a formula specifying the model. For example, if *metadata.vars* is c("var1", "var2"), *combination* could be "var1+var2".

To get the p-values table from the results obtained by using previous function

> multifactor_aov_pvalues_table(multifactor.aov.results, write.file = F, file.out = "multi-anova-pvalues.csv")

- *multifactor.aov.results*: variable containing the multifactor anova results, obtained from using the previous function;
- *write.file*: boolean value (TRUE or FALSE) indicating if you want the results table to be written in a file. Defaults to FALSE;
- *file.out*: string with the file path to write to. You only need to give this information if you chosen to write the results to a file (write.file=T). Defaults to "multi-anova-pvalues.csv".

To get the table with the explained variability from the results obtained by using the function *multifactor_aov_all_vars*

> multifactor_aov_varexp_table(multifactor.aov.results, write.file = F, file.out = "multi-anova-varexp.csv")

- *multifactor.aov.results*: variable containing the multifactor anova results, obtained from using the previous function;
- *write.file*: boolean value (TRUE or FALSE) indicating if you want the results table to be written in a file. Defaults to FAlSE;
- *file.out*: string with the file path to write to. You only need to give this information if you chosen to write the results to a file (write.file=T). Defaults to "multi-anova-varexp.csv".

8.3.2 Examples

Example makes use of an nmr-peaks dataset (NMR peaks dataset used).

1. As the original dataset contains missing values, you must treat the missing values, in order to be able to perform the analysis. The missing values were replaced by the value *0.00005*:

Hide

```
nmr_peaks_dataset_mv=missingvalues_imputation(nmr_peaks_dataset)
```

2. Perform multifactor ANOVA tests on the variables *seasons* and *agroregions* of the dataset, by testing if they are significantly different whith the model "seasons*agroregions":

Hide

```
res_multi_anova_nmr_peaks = multifactor_aov_all_vars(nmr_peaks_dataset_mv, c("seasons","agroregions"), "seasons*agroregions")
```

3. Table with the p-values of the results:

Hide

```
res_multi_anova_nmr_peaks_p.values=multifactor_aov_pvalues_table(res_multi_anova_nmr_peaks)
DT::datatable(res_multi_anova_nmr_peaks_p.values)
```

4. Table with the explained variability of the results:

Hide

```
res_multi_anova_nmr_peaks_exp.var=multifactor_aov_varexp_table(res_multi_anova_nmr_peaks)
DT::datatable(res_multi_anova_nmr_peaks_exp.var)
```

8.4 Kruskal-Wallis Test

8.4.1 Function to use

> kruskalTest_dataset(dataset, metadata.var, threshold = NULL, write.file = F, file.out = "kruskal.csv")

- *dataset*: a specmine dataset;
- *metadata.var*: string representing the metadata variable to use in Kruskal-wallis tests. It is tested the differences in means of the variables between the samples that belong to the different groups of the metadata variable given;
- *treshold*: threshold value of the p-value. Only the results of the variables whose p-values are under the threshold are returned; **optional**
- *write.file*: boolean value (TRUE or FALSE) indicating if you want the results to be written in a file. Defaults to FALSE;
- *file.out*: string with the file path to write to. You only need to give this information if you chosen to write the results to a file (write.file=T). Defaults to "kruskal.csv".

> plot_kruskaltest(dataset, kr.results, kr.threshold = 0.01)

- *dataset*: the specmine dataset that led to the results of the kruskall-wallis tests given in *kr.results* argument;
- *kr.results*: variable containing the kruskal-wallis tests result, obatined from kruskalTest_dataset function;
- *kr.treshold*: numeric value indicating he p-value treshold. An horizontal line will be drawn in the plot and the data points whose p-values are under the treshold will be blue, while the other ones will be grey. Defaults to 0.01.

8.4.2 Examples

Example makes use of an concentrations dataset (concentrations dataset used).

1. Perform Kruskal-Wallis tests on the variables of the dataset, testing if they are significantly different between the classes of the metadata variable *Muscle.loss*. No p-value treshold will be given so that the full results can be seen:

Hide

```
res_kwtest_concentrations = kruskalTest_dataset(concentrations_dataset, "Muscle.loss")
```

2. Table with the results obtained:

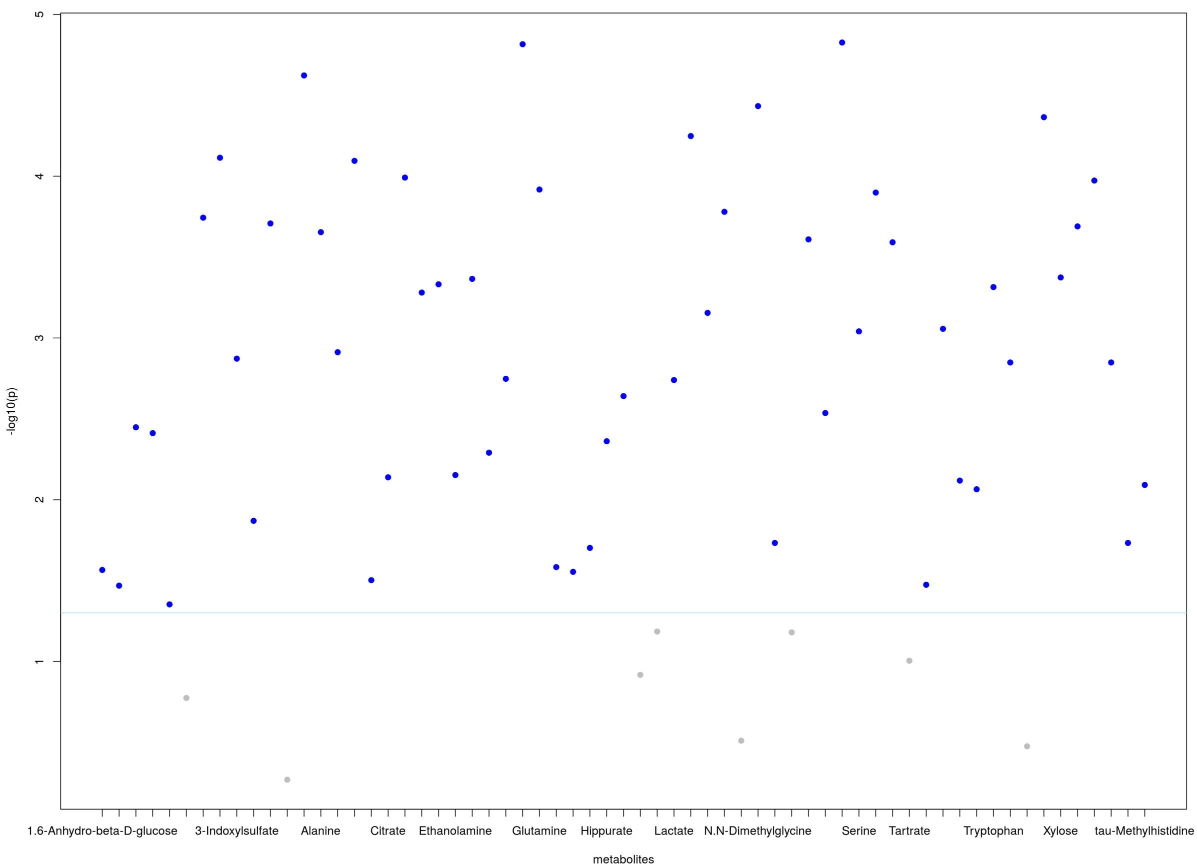
Hide

```
DT::datatable(res_kwtest_concentrations)
```

3. Plot of the results, with a treshold value of 0.05:

Hide

```
plot_kruskaltest(concentrations_dataset, res_kwtest_concentrations, kr.threshold = 0.05)
```



8.5 Kolmogorov Smirnov Test

8.5.1 Function to use

> ksTest_dataset(dataset, metadata.var, threshold = NULL, write.file = F, file.out = "ks.csv")

- *dataset*: a specmine dataset;
- *metadata.var*: string representing the metadata variable to use in Kolmogorov-Smirnov tests. It is tested the differences in means of the variables between the samples that belong to the different groups of the metadata variable given;
- *treshold*: threshold value of the p-value. Only the results of the variables whose p-values are under the threshold are returned; **optional**
- *write.file*: boolean value (TRUE or FALSE) indicating if you want the results to be written in a file. Defaults to FALSE;
- *file.out*: string with the file path to write to. You only need to give this information if you chosen to write the results to a file (write.file=T). Defaults to "ks.csv".

> plot_kstest(dataset, ks.results, ks.threshold = 0.01)

- *dataset*: the specmine dataset that led to the results of the kolmogorov-smirnov tests given in *ks.results* argument;
- *ks.results*: variable containing the kolmogorov-smirnov tests result, obatined from ksTest_dataset function;
- *ks.treshold*: numeric value indicating he p-value treshold. An horizontal line will be drawn in the plot and the data points whose p-values are under the treshold will be blue, while the other ones will be grey. Defaults to 0.01.

8.5.2 Examples

Example makes use of an concentrations dataset (concentrations dataset used).

1. Perform Kolmogorov-Smirnov tests on the variables of the dataset, testing if they are significantly different between the classes of the metadata variable *Muscle.loss*. No p-value treshold will be given so that the full results can be seen:

Hide

```
res_kstest_concentrations = ksTest_dataset(concentrations_dataset, "Muscle.loss")
```

```
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties

## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties

## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties

## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties

## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties

## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties

## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties

## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
```

```
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
```

```
## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties

## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties

## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties

## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties

## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties

## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties

## Warning in ks.test(sub.ds1$data[i, ], sub.ds2$data[i, ]): cannot compute exact p-value with ties
```

2. Table with the results obtained:

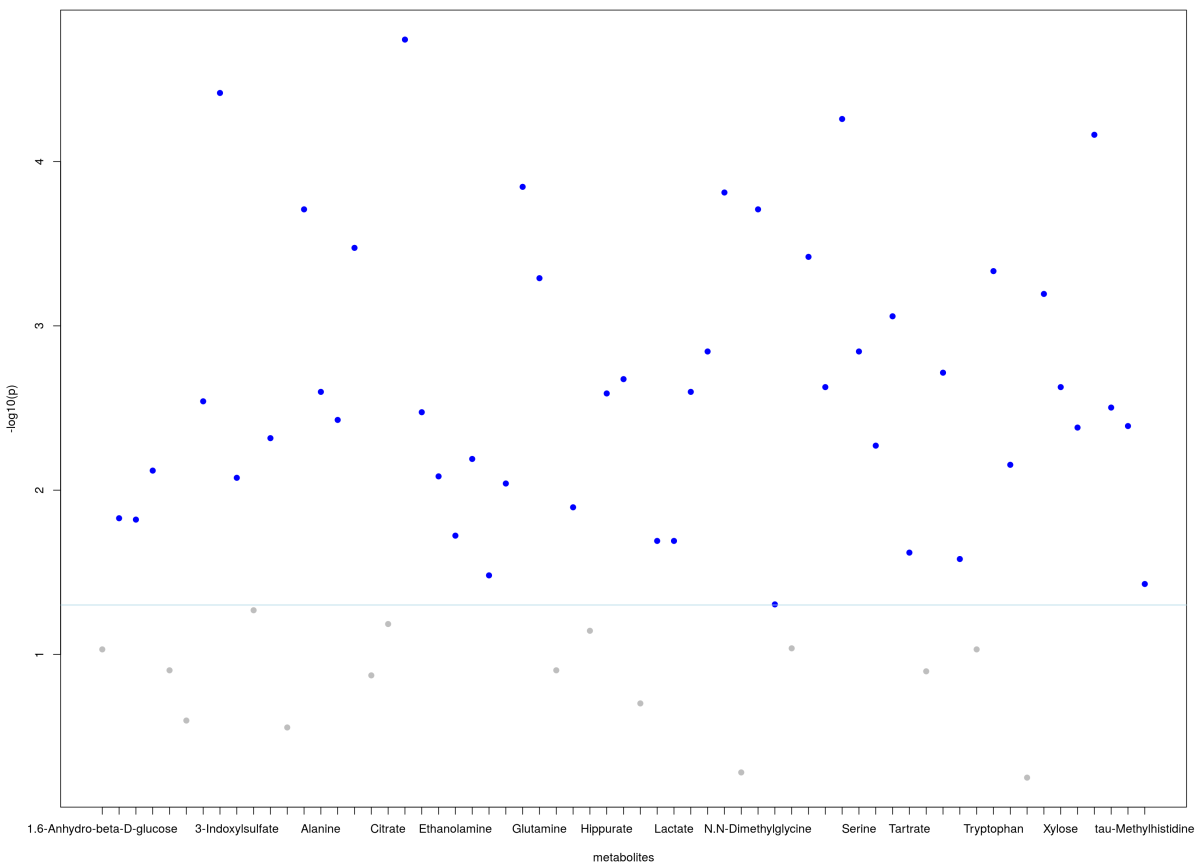
Hide

```
DT::datatable(res_kstest_concentrations)
```

3. Plot of the results, with a p-value treshold of 0.05:

Hide

```
plot_kstest(concentrations_dataset, res_kstest_concentrations, 0.05)
```



8.6 Fold Change Analysis

8.6.1 Functions to use

You can perform fold change in each variable of the dataset, i.e., analyse the difference of each variable on two groups

> fold_change(dataset, metadata.var, ref.value, threshold.min.fc = NULL, write.file = F, file.out = “fold_change.csv”)

- *dataset*: a specmine dataset;
- *metadata.var*: string representing the metadata variable to use in the fold change analysis. It is tested the differences of the variables on the chosen metadata variable classest;
- *ref.value*: string representing the name of one of the classes of the metadata variable chosen that will be considered the initial value;
- *treshold.min.fc*: minimum treshold of the fold change value. Only the results of the groups whose fold change values are above the threshold are returned; **optional**
- *write.file*: boolean value (TRUE or FALSE) indicating if you want the results to be written in a file. Defaults to FALSE;
- *file.out*: string with the file path to write to. You only need to give this information if you chosen to write the results to a file (write.file=T). Defaults to “fold_change.csv”.

You can perform fold change on two variables, i.e., analyse the difference of groups on two variables

> fold_change_var(dataset, metadata.var, variables, threshold.min.fc = NULL, write.file = F, file.out =

> "fold_change_reverse.csv")

- *dataset*: a specmine dataset;
- *metadata.var*: string representing the metadata variable to use in the fold change test. It is tested the differences of the classes in the metadata variable on the two variables chosen in *variables* argument;
- *variables*: character vector with the names representing the two variables being tested;
- *treshold.min.fc*: minimum treshold of the fold change value. Only the results of the groups whose fold change values are above the threshold are returned; **optional**
- *write.file*: boolean value (TRUE or FALSE) indicating if you want the results to be written in a file. Defaults to FALSE;
- *file.out*: string with the file path to write to. You only need to give this information if you chosen to write the results to a file (write.file=T). Defaults to "fold_change_reverse.csv".

You can see a plot of the results

> plot_fold_change(dataset, fc.results, fc.threshold, plot.log = T, var = F, xlab = "")

- *dataset*: the specmine dataset that led to the results of the fold change analysis given in *fc.results* argument;
- *fc.results*: variable containing the fold change analysis result, obatined from fold_change function;
- *fc.treshold*: numeric value indicating he fold change treshold. An horizontal line will be drawn in the plot and the data points whose fold change values are under the treshold will be blue, while the other ones will be grey;
- *plot.log*: boolean value (TRUE or FALSE) indicating if the fold change values are transformed logarithmically or not. Defaults to TRUE;
- *var*: boolean value (TRUE or FALSE) indicating if the label of the xx axis is given in *xlab* argument. Defaults to FALSE;
- *xlab*: string representing the x axis label. **optional**

You can see a plot that joins the results from fold change analysis and t-test analysis

> volcano_plot_fc_tt(dataset, fc.results, tt.results, fc.threshold = 2, tt.threshold = 0.01)

- *dataset*: the specmine dataset that led to the results of the fold change analysis given in *fc.results* argument and the t-test analysis given in *tt.results* argument;
- *fc.results*: variable containing the fold change analysis result, obatined from fold_change function;
- *tt.results*: variable containing the t-test analysis result, obatined from tTests_dataset function;
- *fc.treshold*: numeric value indicating he fold change treshold. A vertical line will be drawn in the plot and the data points whose fold change values are under the treshold will be blue, while the other ones will be grey;
- *tt.treshold*: numeric value indicating he fold change treshold. An horizontal line will be drawn in the plot and the data points whose fold change values are under the treshold will be blue, while the other ones will be grey;

8.6.2 Examples

Example makes use of an concentrations dataset (concentrations dataset used).

1. Perform fold change analysis on each variable of the dataset, testing the difference of each data variable in the two classes (control and cachexic) of the metadata variable *Muscle.loss*. "control" was considered the initial value and no treshold was specified:

Hide

```
res_fc_vars_concentrations = fold_change(concentrations_dataset, "Muscle.loss", ref.value="control")
```

2. Table with the results obtained:

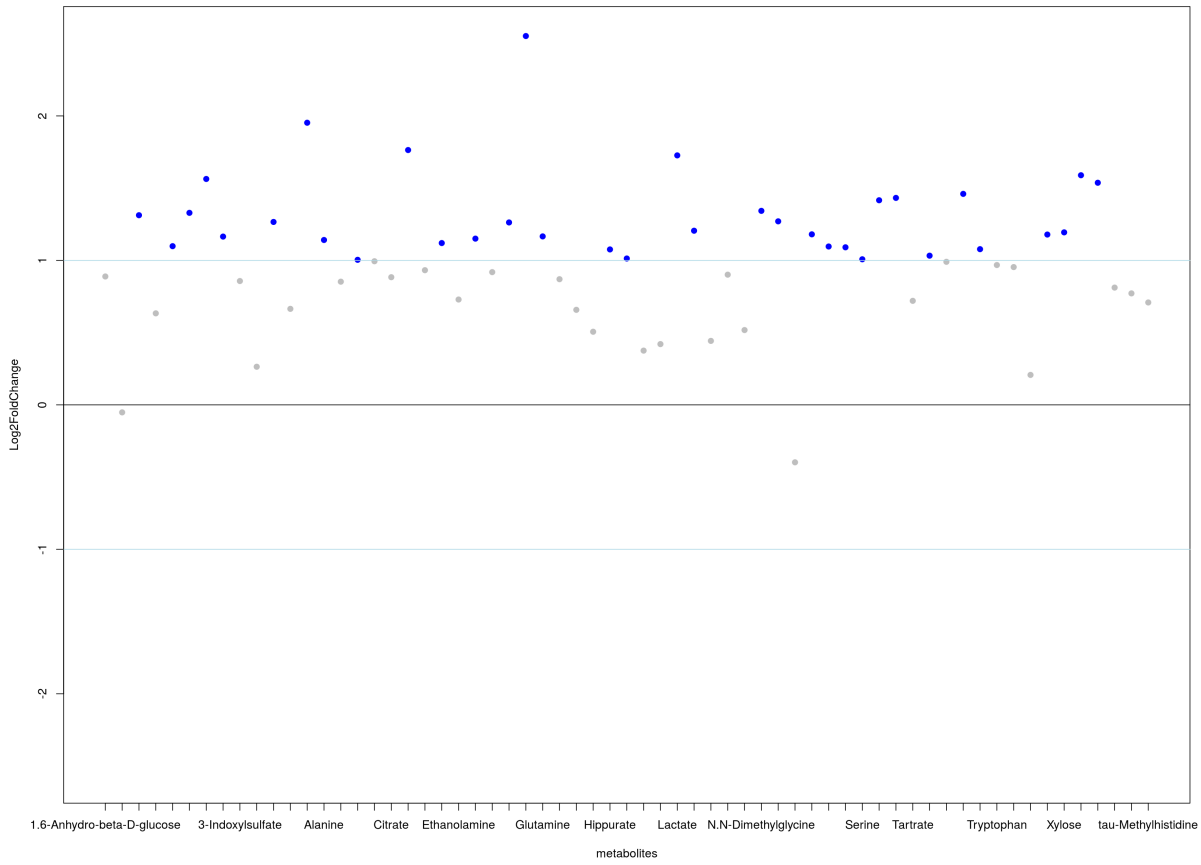
Hide

```
DT::datatable(res_fc_vars_concentrations)
```

3. Plot of the results, with a treshold value of 2:

Hide

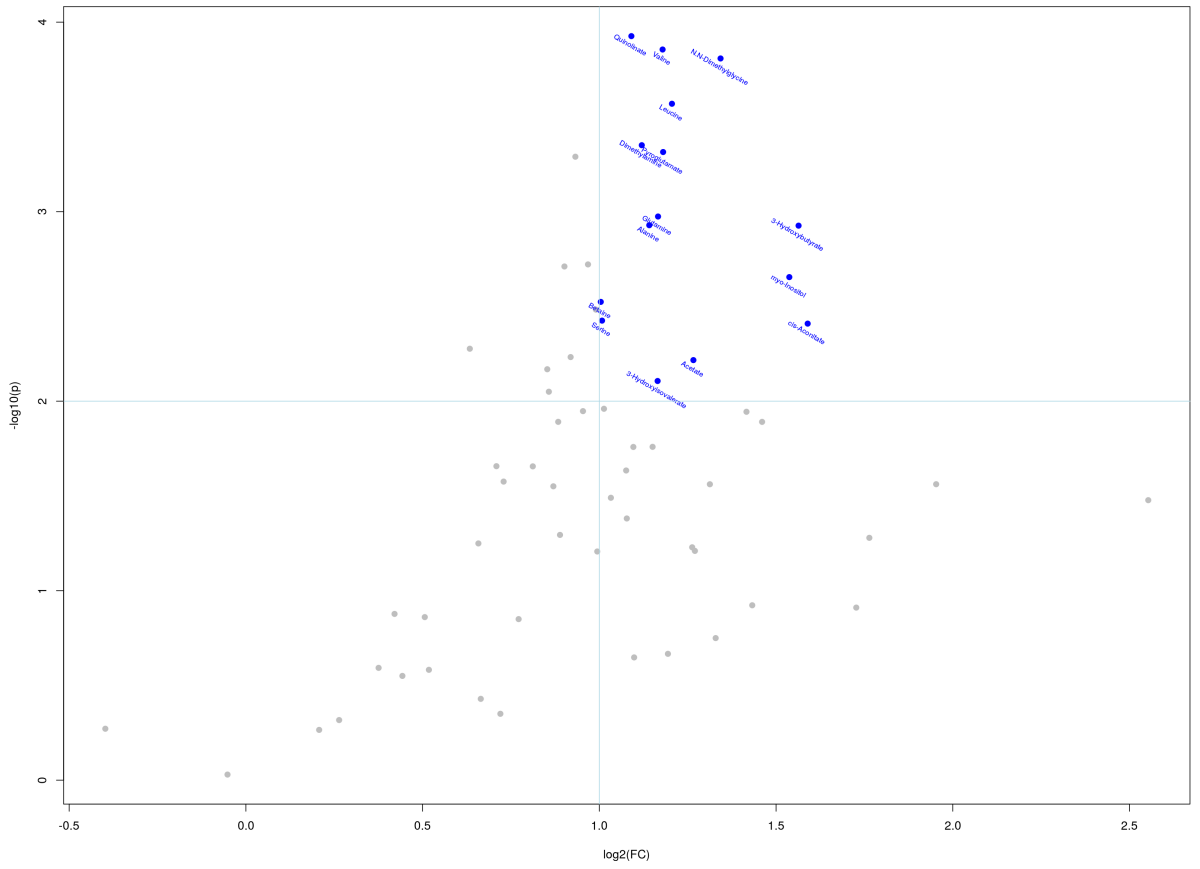
```
plot_fold_change(concentrations_dataset, res_fc_vars_concentrations, 2)
```



4. Plot of the fold change and t-test results (from the example for the t-test analysis), with treshold values of 2 and 0.01 respectively:

Hide

```
volcano_plot_fc_tt(concentrations_dataset, res_fc_vars_concentrations, res_ttest_concentrations)
```



```
##  [1] "3-Hydroxybutyrate"    "3-Hydroxyisovalerate" "Acetate"              "Alanine"              "Betaine"              "Dimethylam
ine"
##  [7] "Glutamine"            "Leucine"              "N.N-Dimethylglycine"  "Pyroglutamate"        "Quinolinate"          "Serine"

## [13] "Valine"               "cis-Aconitate"        "myo-Inositol"
```

5. Perform fold change analysis on the differences of the classes of the *Muscle.loss* metadata variable ("control" and"cachexic") on the variables "Creatine" and "Serine". Again, no treshold was specified:

Hide

```
res_fc_2var_concentrations = fold_change_var(concentrations_dataset, "Muscle.loss", c("Creatine", "Serine"))
```

6. Table with the results obtained:

Hide

```
DT::datatable(res_fc_2var_concentrations)
```

9 Principal Components Analysis (PCA)

9.1 Functions to use

Classical PCA

> pca_analysis_dataset(dataset, scale = T, center = T, write.file = F, file.out = "pca")

- *dataset*: a specmine dataset;
- *scale*: boolean value (TRUE or FALSE) indicating whether the variables should be scaled or not;
- *center*: boolean value (TRUE or FALSE) indicating whether the variables should be centered or not;
- *write.file*: boolean value (TRUE or FALSE) indicating if you want the results to be written in a file. Defaults to FALSE;
- *file.out*: string with the file path to write to. You only need to give this information if you chosen to write the results to a file (write.file=T). Defaults to "pca".

Robust PCA

The difference to the previous method consists on being able to choose how to center and scale the dataset

> pca_robust(dataset, center = "median", scale = "mad", k = 10, write.file = F, file.out = "robpca")

- *dataset*: a specmine dataset;
- *center*: string indicating how variables should be centered. It can either be "median" or "mean". It can also be a numeric vector with the center values of each column (sample). Defaults to "median";
- *scale*: string indicating how variables should be scaled. It can either be standard deviation ratio ("sd") or mean absolute deviation ("mad"). It can also be a numeric vector with the scale value of each column (sample). Defaults to Mean absolute deviation ("mad");
- *k*: number of components to compute;
- *write.file*: boolean value (TRUE or FALSE) indicating if you want the results to be written in a file. Defaults to FALSE;
- *file.out*: string with the file path to write to. You only need to give this information if you chosen to write the results to a file (write.file=T). Defaults to "robpca".

Numerical Results

The following function returns the importance of the principal components:

> pca_importance(pca.res, pcs = 1:length(pca.res$sdev), sd = T, prop = T, cumul = T, min.cum = NULL)

- *pca.res*: variable containing the PCA results, obtained from pca_analysis_dataset or pca_robust function;
- *pcs*: numeric vector indicating for what principal components pca importance should be calculated. Defaults to all components;
- *sd*: boolean value (TRUE or FALSE) indicating whether standard deviation of the results should be returned or not. Defaults to TRUE;
- *prop*: boolean value (TRUE or FALSE) indicating whether proportion of variance of the results should be returned or not. Defauls to TRUE;
- *cumul*: boolean value (TRUE or FALSE) indicating whether the cumulative variance should be returned or not. Defaults to TRUE;
- *min.cum*: numeric value, in percentage, indicating the minimum cumulative percentage of variance. **optional**

9.1.1 Examples

Example makes use of an concentrations dataset (concentrations dataset used).

1. Perform a normal PCA analysis, by scaling and centering the variables:

Hide

```
res_norm_pca_concentrations = pca_analysis_dataset(concentrations_dataset)
```

2. PCA importance table:

Hide

```
res_imp_pca_concentrations=pca_importance(res_norm_pca_concentrations, pcs=1:5)
DT::datatable(res_imp_pca_concentrations, options=list(scrollX=T))
```

3. PCA loadings table:

To get loadings values of robust PCA results, instead of 'rotation' change to 'loadings'

Hide

```
DT::datatable(round(res_norm_pca_concentrations$rotation, 4), options=list(scrollX=T))
```

4. PCA scores table:

To get loadings values of robust PCA results, instead of 'x' change to 'scores'

Hide

```
DT::datatable(round(res_norm_pca_concentrations$x, 4), options=list(scrollX=T))
```

5. To see other results available, get the names of the result's list and then access that result:

_For more details on what the results return see the value section of function prcomp for normal PCA results and princomp for robust PCA results, from stats package.

Hide

```
names(res_norm_pca_concentrations)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

Hide

```
res_norm_pca_concentrations$sdev #Standard deviations of the principal components
```

```
##  [1] 5.04667 2.27013 1.83311 1.74728 1.65906 1.61305 1.47304 1.36403 1.24275 1.20650 1.15841 1.05503 1.03620 0.99140 0.96773 0.8955
1 0.86788 0.83041
## [19] 0.81327 0.73918 0.72112 0.71053 0.64606 0.63389 0.58304 0.54425 0.50539 0.48743 0.42674 0.42427 0.41483 0.38653 0.35092 0.3242
4 0.31646 0.28671
## [37] 0.28435 0.26060 0.25353 0.24800 0.21896 0.19537 0.18914 0.17667 0.16864 0.15799 0.15287 0.13804 0.13101 0.10759 0.10374 0.0985
3 0.08760 0.08258
## [55] 0.08049 0.06927 0.05937 0.05673 0.05088 0.04001 0.02972 0.02789 0.01876
```

9.2 Plot functions and examples

Examples for each plot will use the pca results from the example above

9.2.1 Scree plot

Shows the individual and cumulative percentages of the explained variance of each principal component

> pca_screeplot(pca.result, num.pcs = NULL, cex.leg = 0.8, leg.pos = "right", lab.text = c("individual percent", "cumulative percent"), fill.col = c("blue", "red"), ylab = "Percentage", xlab = "Principal components", ...)

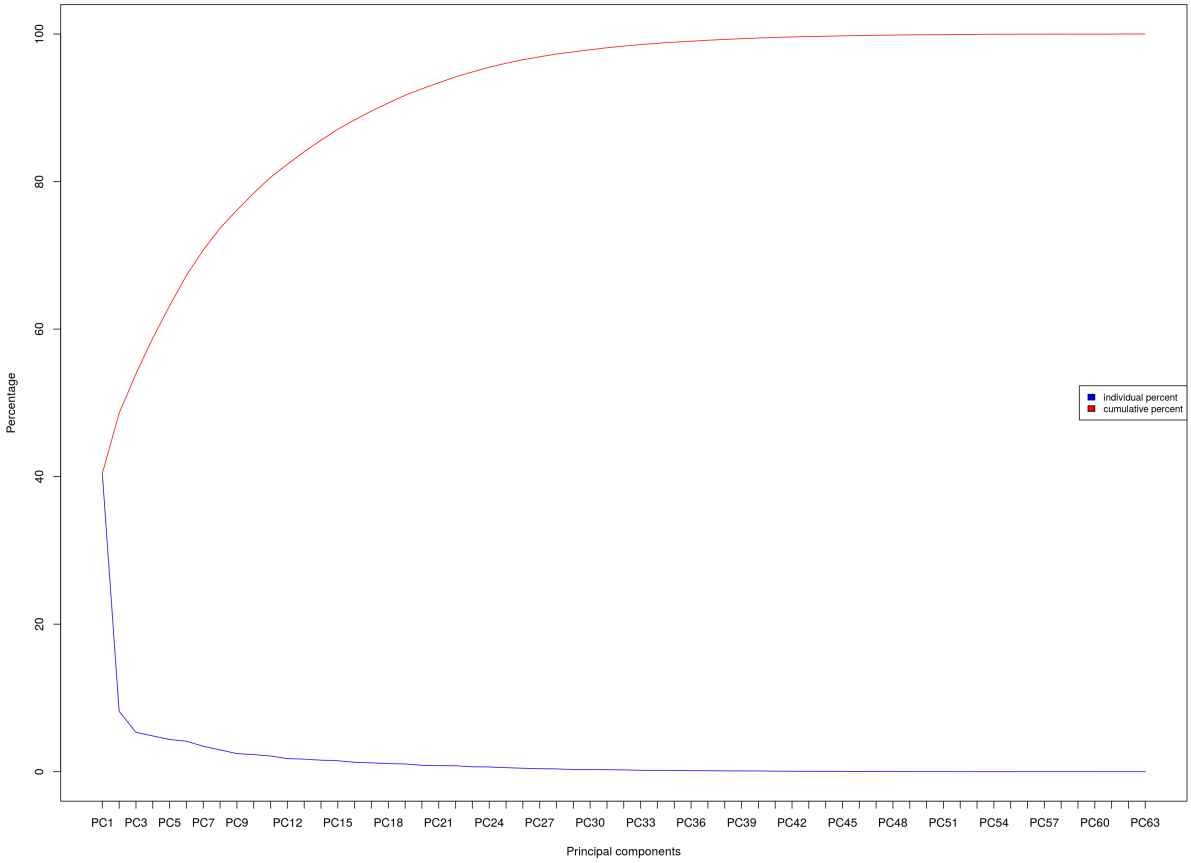
- *pca.result*: variable containing the PCA results, obtained from pca_analysis_dataset or pca_robust function;
- *num.pcs*: number of principal components to show in the plot. If *NULL*, shows all components. Defaults to *NULL*;
- *cex.leg*: number indicating the relative font size of legend. Defaults to 0.8;
- *leg.pos*: string indicating the position of the legend in the plot. It can either be "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or "center". Defaults to "right";
- *lab.text*: character vector with the labels of the legend. Defaults to c("individual percent", "cumulative percent");
- *fill.col*: character vector with the colors of each line. Defaults to c("blue", "red"). Names of the available colors can be consulted using the function [colors]] (https://www.rdocumentation.org/packages/grDevices/versions/3.4.3/topics/colors) from the grDevices package;
- *ylab*: yy axis label. Defaults to "Percentage";
- *xlab*: xx axis label. Defaults to "Principal components";
- ...: aditional parameters of matplot funtion.

9.2.1.1 Examples

1. Plot showing results on all principal components, where the blue line represents the individual percent and the red one the cumulative percent:

Hide

```
pca_screeplot(res_norm_pca_concentrations)
```



9.2.2 Scores plot 2D

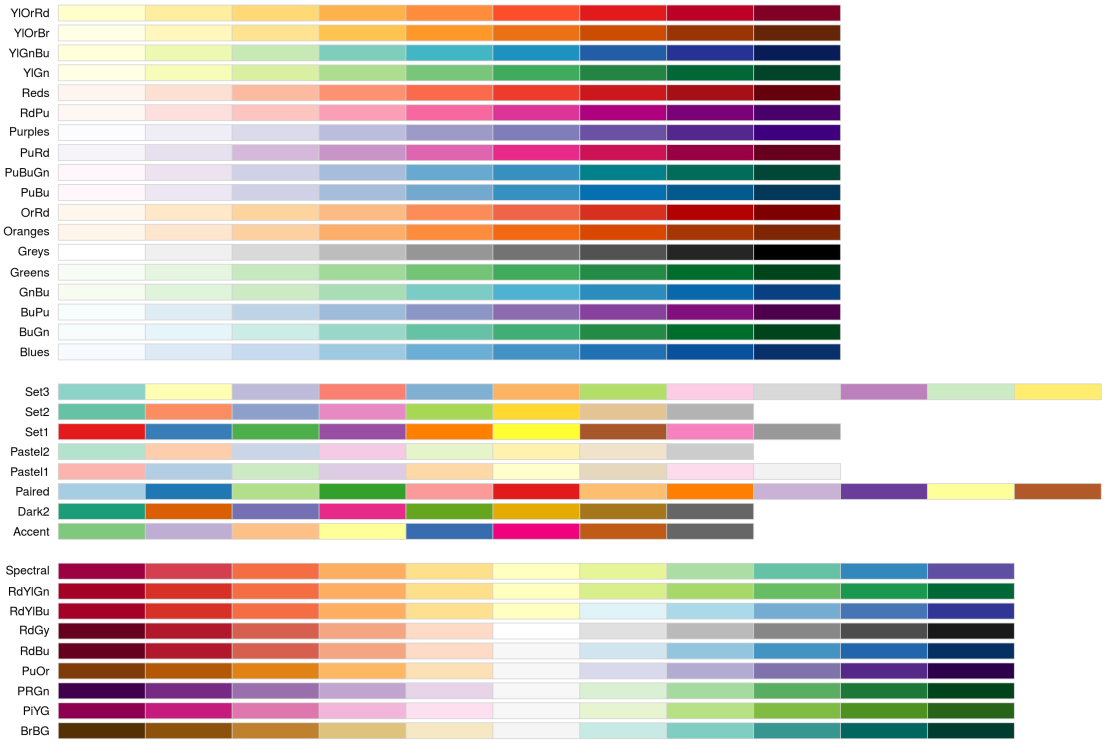
Shows a 2D PCA scores plot of two principal components

> pca_scoresplot2D(dataset, pca.result, column.class, pcas = c(1, 2), labels = FALSE, ellipses = FALSE, bw=F, pallette = 2, leg.pos = "right", xlim = NULL, ylim = NULL)

- *dataset*: the specmine dataset that led to the pca results given in *pca.result* argument;
- *pca.result*: variable containing the PCA results, obtained from pca_analysis_dataset or pca_robust function;
- *column.class*: string representing the name of a metadata variable. The plot is colored by the classes of the metadata variable mentioned in this argument;
- *pcas*: numeric vector with the two principal components to plot. Defaults to c(1,2);
- *labels*: boolean value (TRUE or FALSE) indicating whether samples' names should be shown in the plot (each point in the plot corresponds to a sample) or not. Defaults to FALSE;
- *ellipses*: boolean value (TRUE or FALSE) indicating whether ellipses should be drawn on each class of the metadata variable chosen. Defaults to FALSE. Ellipses are never drawn when *bw* argument is *TRUE*;
- *bw*: boolean value (TRUE or FALSE) indicating whether the plot should black and white or colored. If bw=TRUE, the plot is black and white. Defaults to FALSE;
- *pallette*: pallette of colors by which to color the plot (one different color will be assign to each metadata class). Can be a number or a string indicating the color palette wanted. Examples of color palettes and respective names (from RColorBrewer package):

Hide

```
RColorBrewer::display.brewer.all()
```



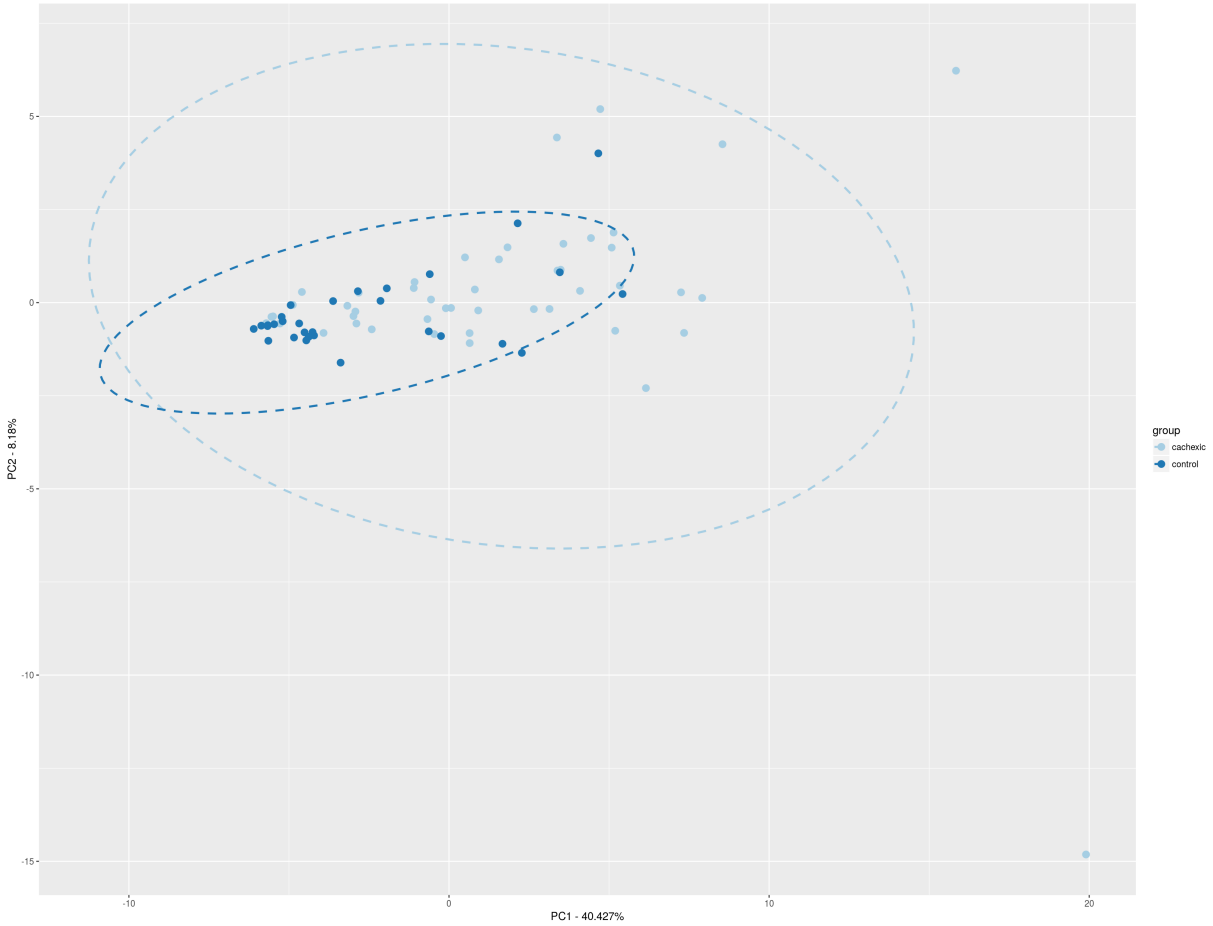
- *leg.pos*: string indicating the position of the legend in the plot. It can either be “bottomright”, “bottom”, “bottomleft”, “left”, “topleft”, “top”, “topright”, “right” or “center”. Defaults to “right”;
- *xlim*: numeric vector indicating the limits of the xx axis. **optional**
- *ylim*: numeric vector indicating the limits of the yy axis. **optional**

9.2.2.1 Examples

1. Colored plot showing results on the first two principal components, with ellipses drawn:

Hide

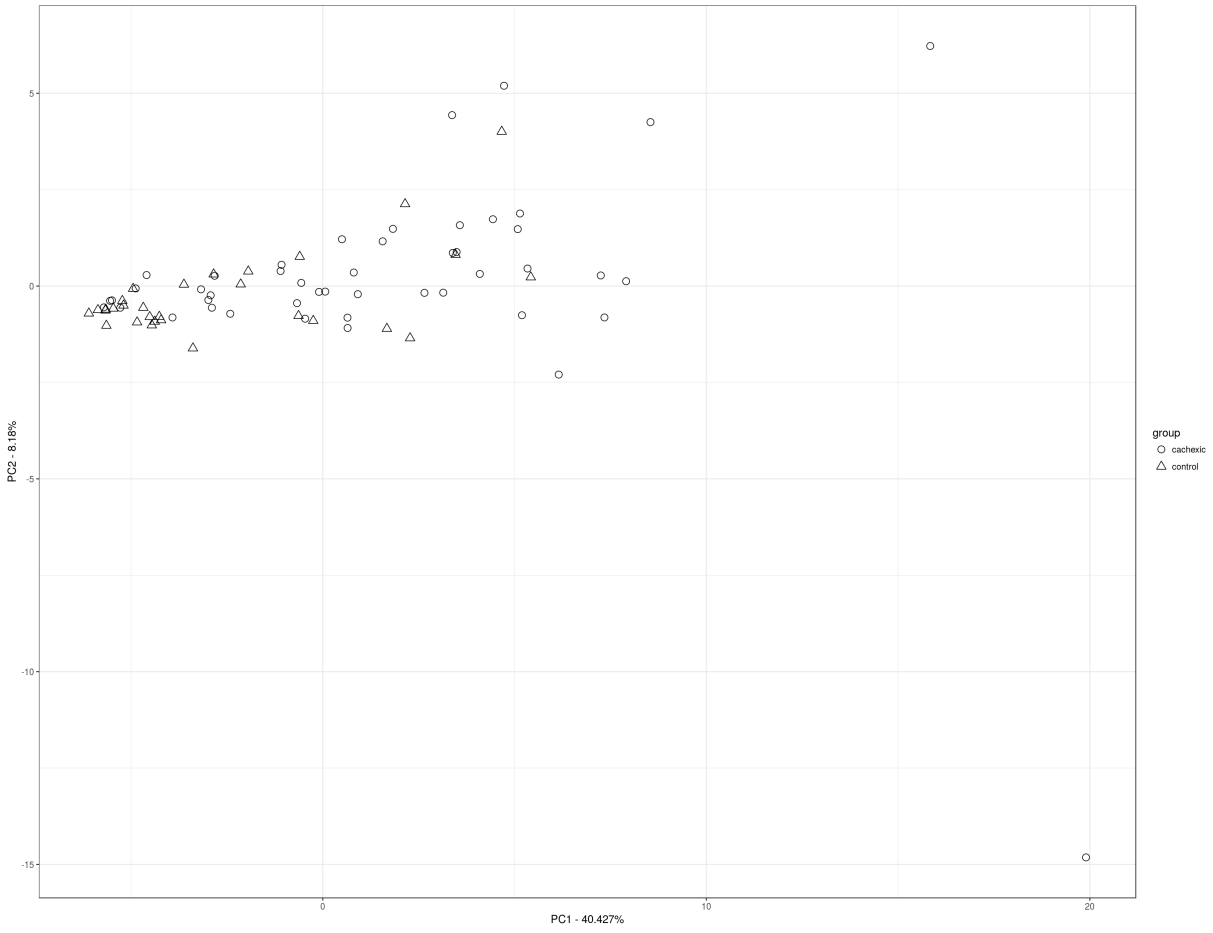
```
pca_scoresplot2D(concentrations_dataset, res_norm_pca_concentrations, "Muscle.loss", pcas = c(1,2), ellipses = TRUE, pallette = "Paired")
```



2. Same plot, but black and white:

Hide

```
pca_scoresplot2D(concentrations_dataset, res_norm_pca_concentrations, "Muscle.loss", bw=TRUE)
```



9.2.3 Normal Scores plot 3D

```
pca_scoresplot3D(dataset, pca.result, column.class, pcas = c(1, 2, 3))
```

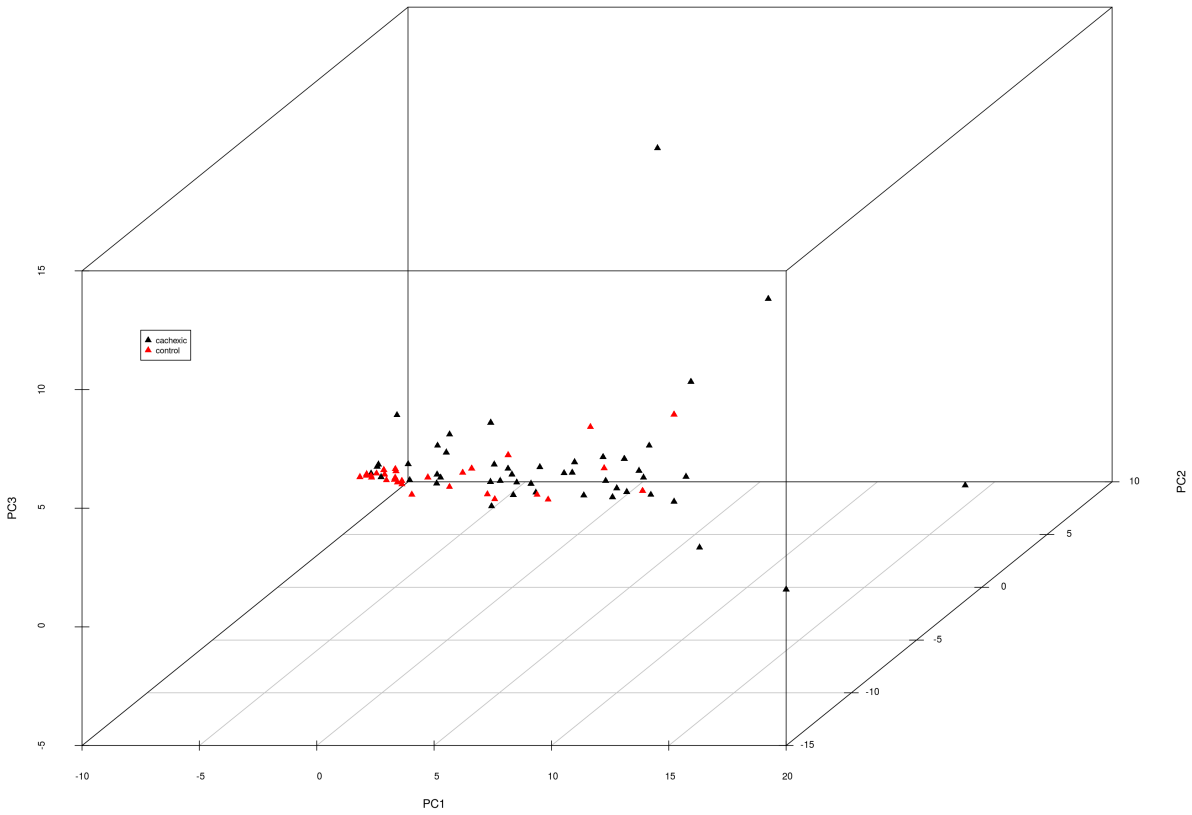
- *dataset*: the specmine dataset that led to the pca results given in *pca.result* argument;
- *pca.result*: variable containing the PCA results, obtained from pca_analysis_dataset or pca_robust function;
- *column.class*: string representing the name of a metadata variable. The plot is colored by the classes of the metadata variable mentioned in this argument;
- *pcas*: numeric vector with the three principal components to plot. Defaults to c(1,2,3).

9.2.3.1 Examples

1. Plot showing results on the first three principal components, colored by the classes of the metadata variable *Muscle.loss*:

Hide

```
pca_scoresplot3D(concentrations_dataset, res_norm_pca_concentrations, "Muscle.loss", pcas = c(1,2,3))
```



9.2.4 Interactive Scores plot 3D

This function uses the rgl package. If having problems, go to section Problems running code: solutions

> pca_scoresplot3D_rgl(dataset, pca.result, column.class, pcas = c(1, 2, 3), size = 1, labels = FALSE)

- *dataset*: the specmine dataset that led to the pca results given in *pca.result* argument;
- *pca.result*: variable containing the PCA results, obtained from pca_analysis_dataset or pca_robust function;
- *column.class*: string representing the name of a metadata variable. The plot is colored by the classes of the metadata variable mentioned in this argument;
- *pcas*: numeric vector with the three principal components to plot. Defaults to c(1,2,3);
- *size*: value indicating the size of the points. Defaults to 1;
- *labels*: boolean value (TRUE or FALSE) indicating whether samples' names should be shown in the plot (each point in the plot corresponds to a sample) or not. Defaults to FALSE.

9.2.4.1 Examples

1. Plot showing results on the first 3 principal components, where each sample is labeled:

When running this, a plot in an interactive rgl window is shown

Hide

```
pca_scoresplot3D_rgl(concentrations_dataset, res_norm_pca_concentrations, "Muscle.loss", labels = TRUE)
```

9.2.5 Biplot 2D

> pca_biplot(dataset, pca.result, cex = 0.8, legend.cex = 0.8, x.colors = 1, inset = c(0, 0), legend.place = "topright", ...)

- *dataset*: the specmine dataset that led to the pca results given in *pca.result* argument;
- *pca.result*: variable containing the PCA results, obtained from pca_analysis_dataset or pca_robust function;
- *cex*: value of the relative font size of legend. Defaults to 0.8;
- *colors*:
- *inset*: argument of the legend function.

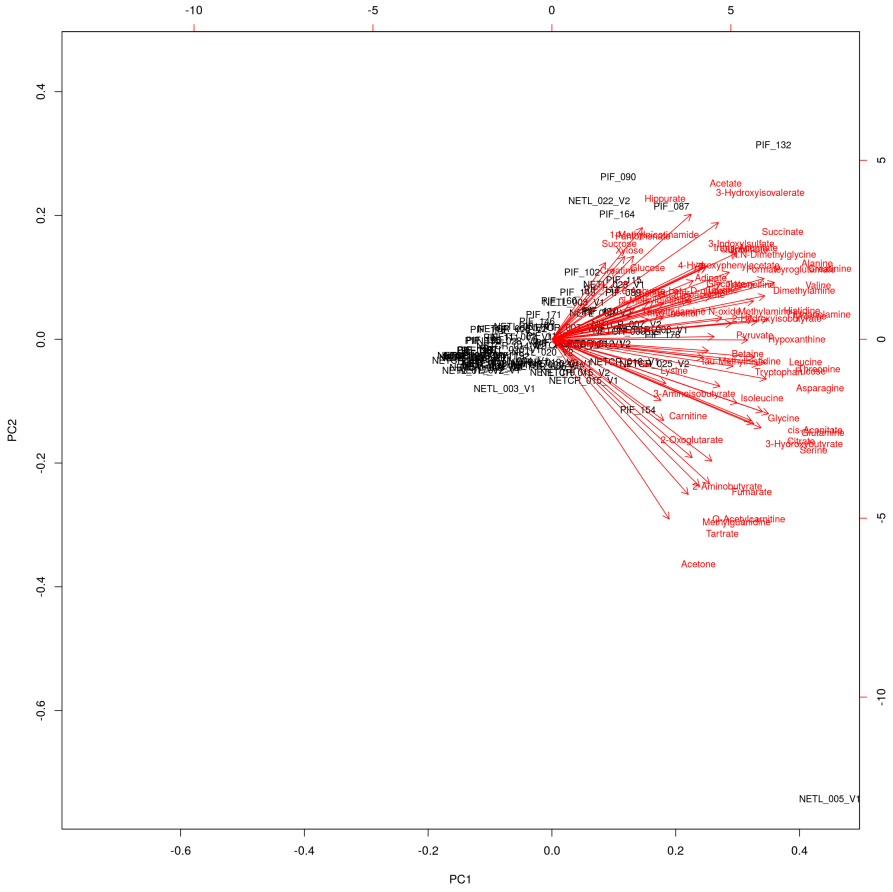
- *legend.place*: string indicating the position of the legend in the plot. It can either be “bottomright”, “bottom”, “bottomleft”, “left”, “topleft”, “top”, “topright”, “right” or “center”. Defaults to “topright”;
- ...: additional parameters for biplot function.

9.2.5.1 Examples

1. Biplot of the first two principal components, colored by the classes in the metadata variable *Muscle.loss*:

Hide

```
pca_biplot(concentrations_dataset, res_norm_pca_concentrations, colors="Muscle.loss")
```



9.2.6 Interactive Biplot 3D

This function uses the rgl package. If having problems, go to section Problems running code: solutions

> pca_biplot3D(dataset, pca.result, column.class = NULL, pcas = c(1, 2, 3))

- *dataset*: the specmine dataset that led to the pca results given in *pca.result* argument;
- *pca.result*: variable containing the PCA results, obtained from pca_analysis_dataset or pca_robust function;
- *column.class*: string representing the name of a metadata variable. The plot is colored by the classes of the metadata variable mentioned in this argument; **optional**
- *pcas*: numeric vector with the three principal components to plot. Defaults to c(1,2,3).

9.2.6.1 Examples

1. Biplot of the first three principal components, colored by the classes in the metadata variable *Muscle.loss*:

When running this, a plot in an interactive rgl window is shown

Hide

```
pca_biplot3D(concentrations_dataset, res_norm_pca_concentrations, "Muscle.loss", pcas = c(1,2,3))
```

9.2.7 Pairs Plot

> pca_pairs_plot(dataset, pca.result, column.class = NULL, pcas = c(1, 2, 3, 4, 5), ...)

- *dataset*: the specmine dataset that led to the pca results given in *pca.result* argument;
- *pca.result*: variable containing the PCA results, obtained from pca_analysis_dataset or pca_robust function;
- *column.class*: string representing the name of a metadata variable. The plot is colored by the classes of the metadata variable mentioned in this argument; **optional**
- *pcas*: numeric vector with the three principal components to plot. Defaults to c(1,2,3);
- ...: additional parameters to ggpairs function.

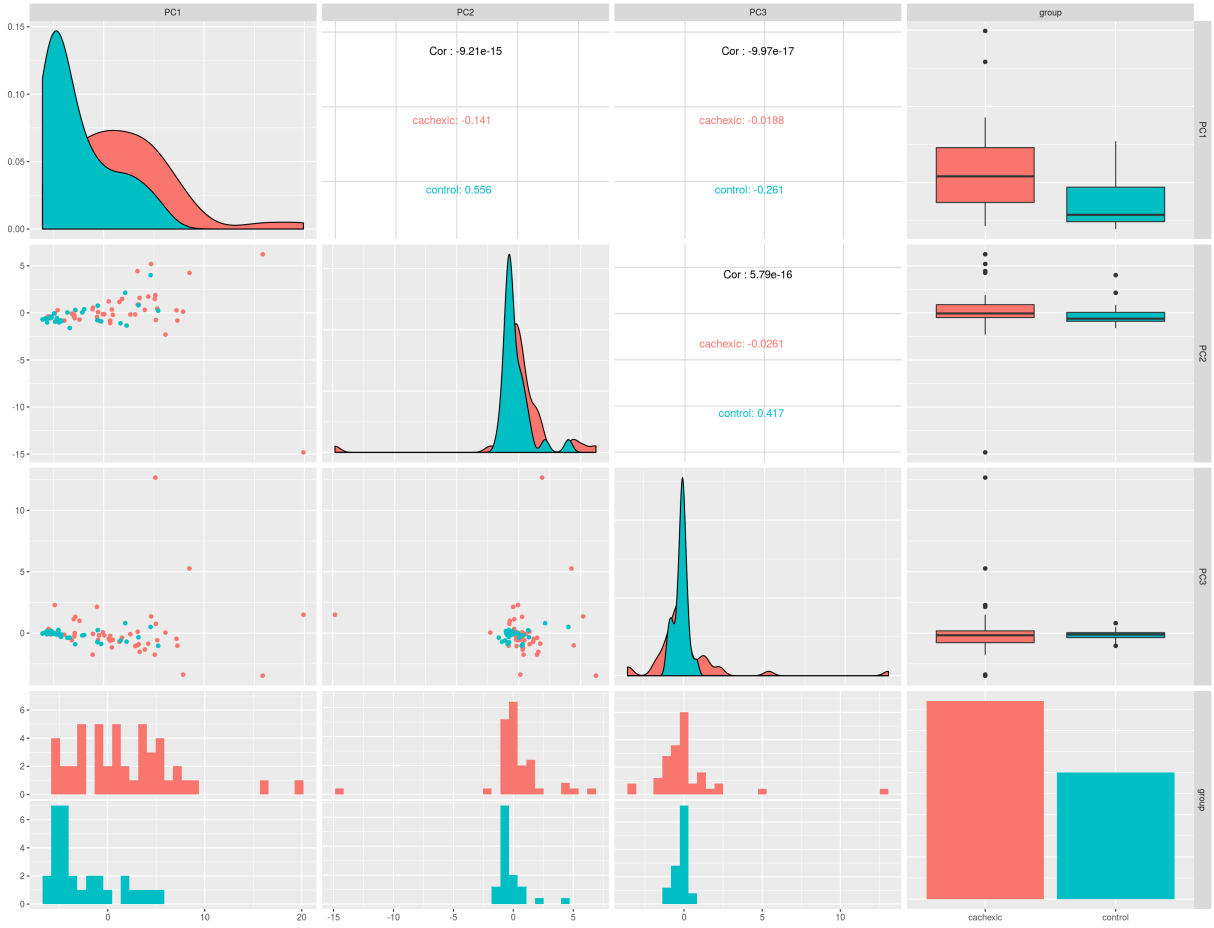
9.2.7.1 Examples

1. Pairs plot for the first three principal components, colored by the classes in the metadata variable *Muscle.loss*:

Hide

```
pca_pairs_plot(concentrations_dataset, res_norm_pca_concentrations, "Muscle.loss", pcas = c(1,2,3))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



9.2.8 Kmeans Plot 2D

> pca_kmeans_plot2D(dataset, pca.result, num.clusters = 3, pcas = c(1, 2), kmeans.result = NULL, labels = FALSE, bw=F, ellipses = FALSE, leg.pos = "right", xlim = NULL, ylim = NULL)

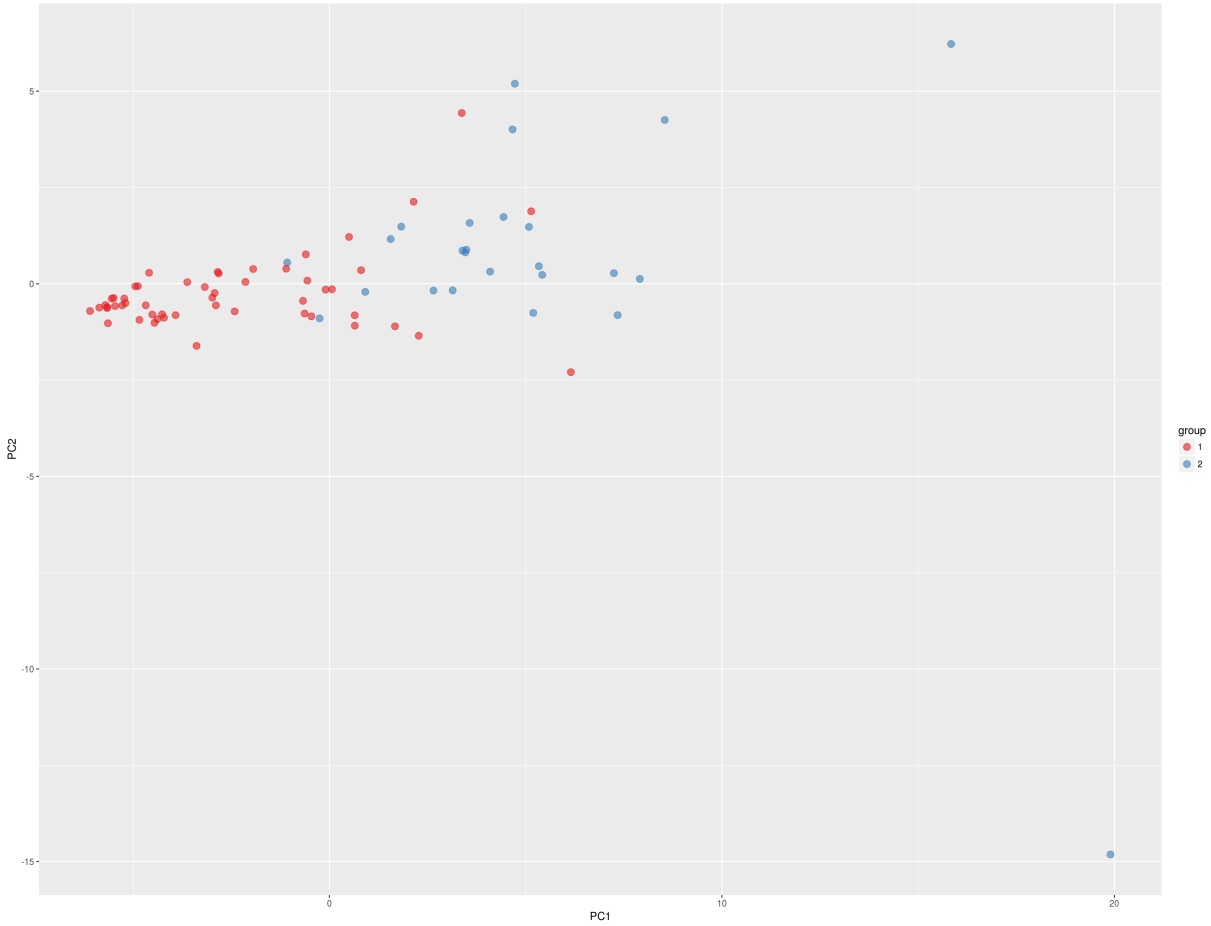
- *dataset*: the specmine dataset that led to the pca results given in *pca.result* argument;
- *pca.result*: variable containing the PCA results, obtained from pca_analysis_dataset or pca_robust function;
- *num.clusters*: value indicating the number of clusters to set to when performing the k-means. Only relevant if kmeans.result=NULL;
- *pcas*: numeric vector with the three principal components to plot. Defaults to c(1,2,3);
- *kmeans.result*: variable containing a k-means result. If not given (kmeans.result=NULL), k-means is performed by the function;
- *labels*: boolean value (TRUE or FALSE) indicating whether samples' names should be shown in the plot (each point in the plot corresponds to a sample) or not. Defaults to FALSE;
- *ellipses*: boolean value (TRUE or FALSE) indicating whether ellipses should be drawn on each class of the metadata variable chosen. Defaults to FALSE. Ellipses are never drawn when *bw* argument is *TRUE*;
- *bw*: boolean value (TRUE or FALSE) indicating whether the plot should black and white or colored. If bw=TRUE, the plot is black and white. Defaults to FALSE;
- *leg.pos*: string indicating the position of the legend in the plot. It can either be "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or "center". Defaults to "right";
- *xlim*: numeric vector indicating the limits of the xx axis. **optional**
- *ylim*: numeric vector indicating the limits of the yy axis. **optional**

9.2.8.1 Examples

1. Kmeans plot for the first two components, where a k-means of two clusters is performed:

Hide

```
pca_kmeans_plot2D(concentrations_dataset, res_norm_pca_concentrations, num.clusters = 2)
```



9.2.9 Kmeans Plot 3D

This function uses the rgl package. If having problems, go to section Problems running code: solutions

> pca_kmeans_plot3D(dataset, pca.result, num.clusters = 3, pcas = c(1, 2, 3), kmeans.result = NULL, labels = FALSE, size = 1, ...)

- *dataset*: the specmine dataset that led to the pca results given in *pca.result* argument;
- *pca.result*: variable containing the PCA results, obtained from pca_analysis_dataset or pca_robust function;
- *num.clusters*: value indicating the number of clusters to set to when performing the k-means. Only relevant if kmeans.result=NULL;
- *pcas*: numeric vector with the three principal components to plot. Defaults to c(1,2,3);
- *kmeans.result*: variable containing a k-means result. If not given (kmeans.result=NULL), k-means is performed by the function;
- *labels*: boolean value (TRUE or FALSE) indicating whether samples' names should be shown in the plot (each point in the plot corresponds to a sample) or not. Defaults to FALSE;
- *size*: value indicating the size of the points. Defaults to 1.

9.2.9.1 Examples

1. Kmeans plot for the first three components, where a k-means of two clusters is performed:

When running this, a plot in an interactive rgl window is shown

Hide

```
pca_kmeans_plot3D(concentrations_dataset, res_norm_pca_concentrations, num.clusters = 2, pcas = c(1,2,3))
```

9.2.10 Kmeans Pairs Plot

> pca_pairs_kmeans_plot(dataset, pca.result, num.clusters = 3, kmeans.result = NULL, pcas = c(1, 2, 3, 4, 5))

- *dataset*: the specmine dataset that led to the pca results given in *pca.result* argument;
- *pca.result*: variable containing the PCA results, obtained from pca_analysis_dataset or pca_robust function;
- *num.clusters*: value indicating the number of clusters to set to when performing the k-means. Only relevant if kmeans.result=NULL;
- *kmeans.result*: variable containing a k-means result. If not given (kmeans.result=NULL), k-means is performed by the function;
- *pcas*: numeric vector with the three principal components to plot. Defaults to c(1,2,3).

9.2.10.1 Examples

1. Kmeans plot for the first three components, where a k-means of two clusters is performed:

Hide

```
pca_pairs_kmeans_plot(concentrations_dataset, res_norm_pca_concentrations, num.clusters = 2, pcas = c(1,2,3,4,5))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

10 Clustering Analysis

10.1 Function to use

> clustering(dataset, method = “hc”, distance = “euclidean”, type = “samples”, num.clusters = 5, clustMethod = “complete”)

- *dataset*: a specmine dataset;
- *method*: a string representing the clustering method. It can either be:
  - “hc”: Hierarchical Clustering. Default value;
  - “kmeans”: K-Means Clustering
- *distance*: only necessary for hierarchical clustering, a string representing the distance measure to be used when computing the distances between the samples or variables. It can either be “euclidean”, “manhattan”, “pearson” or “spearman”. Defaults to “euclidean”;
- *type*: a string representing if the cluster analysis should be performed on samples (“samples”) or on variables (“variables”). Defaults to “samples”;
- *num.clusters*: only necessary for k-means clustering, a numeric value indicating the number of clusters in the k-means analysis. Defaults to 5;
- *clustMethod*: only necessary for hierarchical clustering, a string representing the agglomeration method. It can either be “complete”, “ward”, “single”, “average”, “mcquitty”, “median” or “centroid”. Defaults to “complete”.

10.2 Hierarchical Clustering

10.2.1 Results functions

> dendrogram_plot_col(dataset, hc.result, classes.col, colors = NULL, title = “”, lab.cex = 1, leg.pos = “topright”, label_samples=NULL)

- *dataset*: the specmine dataset that led to the results of hierarchical clustering given in *hc.result* argument;
- *hc.clust*: variable containing the hierarchical clustering result, obtained from clustering function;
- *classes.col*: string or index representing the metadata variable by which the leafs will be colored;
- *colors*: vector with the different colors for each metadata class in the metadata variable chosen in *classes.col* argument. If nothing is given (colors=NULL), the colors will be generated automatically. Defaults to NULL;
- *title*: string representing the title that should appear at the top of the plot; **optional**
- *lab.cex*: numeric value representing the realtive size of the labels’ text. Defaults to 1;
- *leg.pos*: string representing the position of the legend in the plot. It can either be “bottomright”, “bottom”, “bottomleft”, “left”, “topleft”, “top”, “topright”, “right” or “center”. Defaults to “right”;
- *label_samples*: string or index representing the metadata variable by which the leafs will be named. If not given (label_samples=NULL), the leafs are named

with the respective samples names. Defaults to NULL.

10.2.2 Examples

Example makes use of an concentrations dataset (concentrations dataset used).

1. Hierarchical clustering of the samples, using the euclidean distance and the agglomeration method complete:

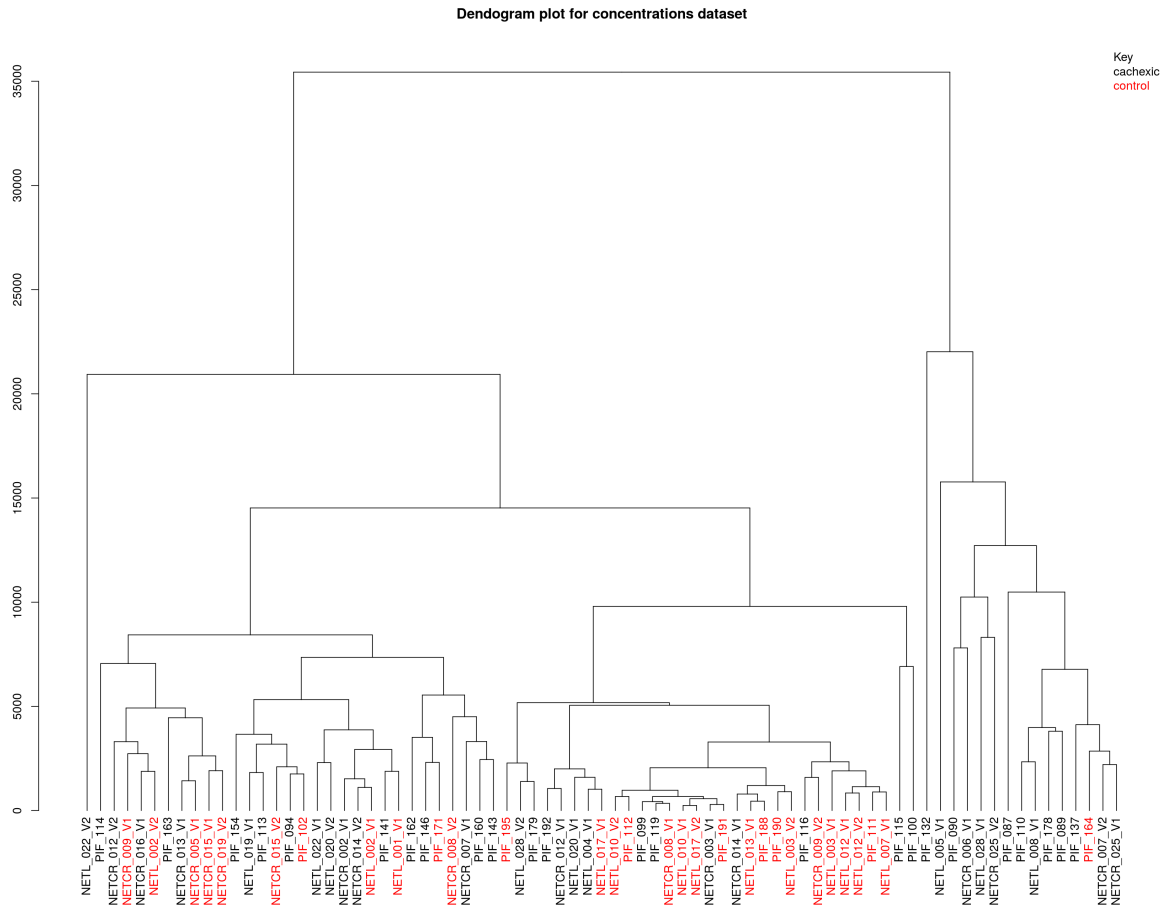
Hide

```
res_hclust_concentrations = clustering(concentrations_dataset)
```

2. Dendogram plot, where samples are colored by the metadata variable *Muscle.loss*:

Hide

```
dendrogram_plot_col(concentrations_dataset, res_hclust_concentrations, "Muscle.loss", title = "Dendogram plot for concentrations datas
et")
```



3. The hierarchical clustering result obtained is a list with the following components:

To know more about the returned results for a hierarchical clustering, see the information on the function hclust.

Hide

```
names(res_hclust_concentrations)
```

```
## [1] "merge"       "height"      "order"       "labels"      "method"      "call"        "dist.method"
```

10.3 K-Means Clustering

10.3.1 Results functions

You can visualize a k-means plot for each cluster

> kmeans_plot(dataset, kmeans.result)

- *dataset*: the specmine dataset that led to the results of k-means clustering given in *hc.result* argument;
- *kmeans.result*: variable containing the k-means clustering result, obtained from clustering function;

You can see a dataframe with the samples that belong to each cluster

> kmeans_result_df(kmeans.result)

- *kmeans.result*: variable containing the k-means clustering result, obtained from clustering function;

10.3.2 Examples

Example makes use of an concentrations dataset (concentrations dataset used).

1. Hierarchical clustering of the samples, using the euclidean distance and the agglomeration method complete:

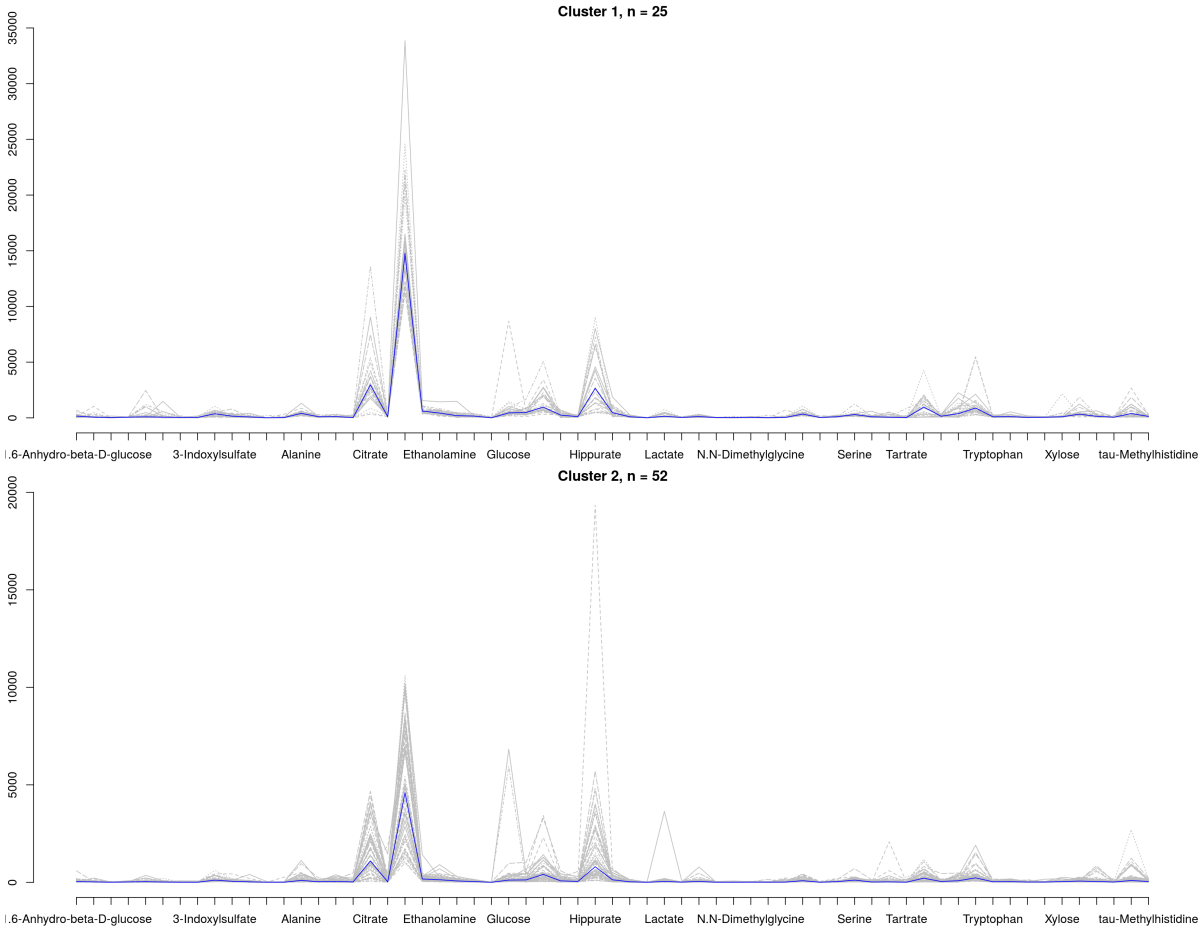
Hide

```
res_kclust_concentrations = clustering(concentrations_dataset, method="kmeans", num.clusters=2)
```

2. K-means results plot, where the grey represents the values of all samples of that cluster and blue the median of those samples:

Hide

```
kmeans_plot(concentrations_dataset, res_kclust_concentrations)
```



3. K-means results data frame:

Hide

```
DT::datatable(kmeans_result_df(res_kclust_concentrations), options=list(scrollX=T), rownames=F)
```

4. The k-means clustering result obtained is a list with the following components:

To know more about the returned results for a hierarchical clustering, see the information on the function kmeans.

Hide

```
names(res_kclust_concentrations)
```

```
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault
"
```

11 Machine Learning

Machine learning in this package is only available for classification problems. Try to convert the numeric metadata to factors so that you can use them for machine learning

11.1 Train Models

> train_models_performance(dataset, models, column.class, validation, num.folds = 10, num.repeats = 10, tunelength = 10, tunegrid = NULL, metric = NULL, compute.varimp = T)

- *dataset*: a specmine dataset;
- *models*: character vector indicating the models to train. The models that can be used are the ones available for the *caret* package. A list with the available models can be accessed here. The names to use in this argument that represent the model wanted are in the column *"method Value"*. Only classification models can be used (*"Type"* column);
- *column.class*: string representing the metadata variable to predict;
- *validation*: string representing the validation method. It can either be:
  - "boot": Resampling;
  - "cv": Cross-Validation;
  - "repeatedcv": Repeated Cross-validation;
  - "loocv": Leave One Out Cross-Validation;
  - "lgocv": Leave Group Out Cross-Validation.

- *num.folds*: if a cross validation method is chosen, a numeric value with the number of folds must be given. Defaults to 10;
- *num.repeats*: if "boot" or "repeatedcv" are chosen, a numeric value with the number of repeats must be given. Defaults to 10;
- *tunelength*: numeric value indicating the number of different values for each parameter of the model(s) chosen should be tested, to find the best one among them. Defaults to 10;
- *tunegrid*: list of dataframe(s) with possible tuning values for each model to train. See examples below for more details (*step 8*);
- *metric*: string indicating the metric to use when evaluating the model's performance. It can either be "Accuracy" or "ROC";
- *compute.varimp*: boolean value (TRUE or FALSE) indicating whether variable importance should be calculated (compute.varimp=TRUE) or not (compute.varimp=FALSE). Defaults to TRUE.

3D Plot for a final PLS model trained

> pca_plot_3d(dataset, model, var.class, pcas = 1:3, colors = NULL, legend.place = "topright", ...)

- *dataset*: the specmine dataset that led to the final PLS model trained given in *model* argument;
- *model*: variable containing a pls final model, obtained from the results of train_models_performance function, in 'results\(final.model\)pls';
- *var.class*: string representing the name of the metadata variable by which to color the plot data points;
- *pcas*: numeric vector with the 3 components to plot. Defaults to c(1,2,3), the first three components of the model;
- *colors*: character vector with the names of the colors that will represent each class of the metadata variable;
- *legend.place*: string indicating the position of the legend in the plot. It can either be "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or "center". Defaults to "topright";
- ...: additional parameters of legend function.

11.1.1 Examples

Example makes use of an concentrations dataset (concentrations dataset used).

1. Training model PLS ("pls") and Random Forests ("rf") to predict the metadata variable *Muscle.loss* (i.e., if the sample belongs to *cachexic* or *control*). For model validation, it was here used the cross validation method ("cv"), with 10 folds. For parameter optimization, it was set to test 10 different values of each parameter of the two models chosen:

Hide

```
res_train_concentrations=train_models_performance(concentrations_dataset, c("pls", "rf"), "Muscle.loss", "cv", metric="Accuracy")
```

```
## Warning: `repeats` has no meaning for this resampling method.
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:specmine':
##
##     multiClassSummary
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:caret':
##
##     R2
```

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

```
## Warning: `repeats` has no meaning for this resampling method.
```

2. Table with the performances of each best model trained:

Hide

```
DT::datatable(res_train_concentrations$performance)
```

3. The parameters of each best model trained:

- PLS model

Hide

```
DT::datatable(res_train_concentrations$best.tunes$pls)
```

- Random Forest model

Hide

```
DT::datatable(res_train_concentrations$best.tunes$rf)
```

4. Tables with the variable importance for each model:

- PLS model

Hide

Hide

```
DT::datatable(res_train_concentrations$vips$pls)
```

- Random Forest model

Hide

```
DT::datatable(res_train_concentrations$vips$rf)
```

5. Results for each value of the parameters tested:

- PLS model

Hide

```
DT::datatable(res_train_concentrations$full.results$pls)
```

- Random Forest model

Hide

```
DT::datatable(res_train_concentrations$full.results$rf)
```

6. Confusion matrices for each model trained:

- PLS model

Hide

```
res_train_concentrations$confusion.matrices$pls
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction cachexic control
##   cachexic     49.4    16.9
##   control      11.7    22.1
##
##  Accuracy (average) : 0.7143
```

- Random Forest model

Hide

```
res_train_concentrations$confusion.matrices$rf
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction cachexic control
##   cachexic     49.4    15.6
##   control      11.7    23.4
##
##  Accuracy (average) : 0.7273
```

7. Each final model can be accessed through `res_train_concentrations$final.models$pls` for the PLS model and `res_train_concentrations$final.models$rf` for the Random Forests model. It is with these final models that the user can predict new samples.

8. If you want to provide specific parameter values to test, this is how you use the *tungrid* argument. You will have to consult the documentation in caret package to know what are the parameters than can be tested for each model.

- Here, it is present an example for the most commonly used models on how to create the dataframe of tuning parameters for each one of them, using *expand.grid* function:

Hide

```
##VALUES HERE INTRODUCED FOR PARAMETER OPTIMIZATION ARE ONLY FOR EXPLAINING PURPOSES, THEY MIGHT NOT BE THE BEST VALUES FOR YOU TO US
E IN TESTING###

##For PLS model - "pls":
#####This models only has one parameter - ncomp
nComp=c(1,2,3,4,5,6) #Here we are saying that we want to test the values from 1 to 6 for this parameter
pls_tuningValues_dataframe=expand.grid(ncomp=nComp)

##For DECISION TREE (C4.5-like Trees) model - "J48":
#####This model has two parameters - M and C
m=c(1,2,3)
C=c(1,2,3,4)
J48_tuningValues_dataframe=expand.grid(M=m, C=C)

##For RULE BASED CLASSIFIER model - "JRip":
#####This model has theree parameters - NumOpts, NumFolds and MinWeights
numopts=c(1,2,3)
numfolds=c(1,2,3)
minweights=c(1,2,3)
JRip_tuningValues_dataframe=expand.grid(NumOpt=numopts, NumFolds=numfolds, MinWeights=minweights)

##For SVM LINEAR model - "svmLinear":
#####This model has one parameter - cost
cost=c(1,4,6)
svmLinear_tuningValues_dataframe=expand.grid(cost=cost)

##For RANDOM FORESTS model - "rf":
#####This model has one parameter - mtry
mtry=c(5,7,8)
rf_tuningValues_dataframe=expand.grid(mtry=mtry)

##For NEURAL NETWORKS model - "nnet":
#####This model has two parameters - size and decay
size=c(3,4,5)
decay=c(1,2)
nnet_tuningValues_dataframe=expand.grid(size=size, decay=decay)
```

- Now, if wanted to train PLS and Random Forests models, like the example above, but using specific parameters to test, you should perform as follows:

Hide

```
#Creating the list of dataframes of tuning parameters
tuneList=list()
tuneList[["pls"]]=pls_tuningValues_dataframe
tuneList[["rf"]]=rf_tuningValues_dataframe

res_train_concentrations=train_models_performance(concentrations_dataset, c("pls", "rf"), "Muscle.loss", "cv", metric="Accuracy", tune
grid=tuneList)
```

```
## Warning: `repeats` has no meaning for this resampling method.

## Warning: `repeats` has no meaning for this resampling method.
```

- As you can see, now the results for each value of the parameters tested are:

PLS model

Hide

```
DT::datatable(res_train_concentrations$full.results$pls)
```

Random Forest model

Hide

```
DT::datatable(res_train_concentrations$full.results$rf)
```

11.2 Predict new samples

> predict_samples(train.result, new.samples)

- *train.result*: variable containing a final model obtained from the function train_models_performance;
- *new.samples*: the data matrix from a specmine dataset, where it will be predicted, for each sample, the value of the metadata variable trained to predict in the model given in *train.result* argument.

11.2.1 Example

Example makes use of an concentrations dataset (concentrations dataset used) and the results obtained in the example for the function train_models_performance.

As we do not have samples whose output is unkown for these or other datasets here being used, we will here use the same data used to train the models (only for explaining purposes).

1. Here, we use the PLS model to predict the ouput of the samples given:

Hide

```
res_predict_concentrations=predict_samples(res_train_concentrations$final.models$pls, concentrations_dataset$data)
```

2. Table with the results. The first column contains the name of each sample predicted and the second column the output value given:

Hide

```
DT::datatable(res_predict_concentrations)
```

11.3 Train and Predict

you can train models and predict new samples in only one function

> train_and_predict(dataset, new.samples, column.class, model, validation, num.folds = 10, num.repeats = 10, tunelength = 10, tunegrid = NULL, metric = NULL)

- *dataset*: a specmine dataset;
- *new.samples*: a specmine dataset, where it will be predicted, for each sample, the value of the metadata variable trained to predict in the model given in *model* argument;
- *column.class*: string representing the metadata variable to predict;
- *model*: string representing the model to train. The models that can be used are the ones available for the *caret* package. A list with the available models can be accessed here. The names to use in this argument that represent the model wanted are in the column *"method Value"*. Only classification models can be used (*"Type"* column);
- *validation*: string representing the validation method. It can either be:
  - "boot": Resampling;
  - "cv": Cross-Validation;
  - "repeatedcv": Repeated Cross-validation;
  - "loocv": Leave One Out Cross-Validation;
  - "lgocv": Leave Group Out Cross-Validation.
- *num.folds*: if a cross validation method is chosen, a numeric value with the number of folds must be given. Defaults to 10;
- *num.repeats*: if "boot" or "repeatedcv" are chosen, a numeric value with the number of repeats must be given. Defaults to 10;
- *tunelength*: numeric value indicating the number of different values for each parameter of the model(s) chosen should be tested, to find the best one among them. Defaults to 10;
- *tunegrid*: list of dataframe(s) with possible tuning values for each model to train. See examples below for more details (*step 8*);
- *metric*: string indicating the metric to use when evaluating the model's performance. It can either be "Accuracy" or "ROC".

12 Feature Selection

12.1 Function to use

> feature_selection(dataset, column.class, method = "rfe", functions, validation = "cv", repeats = 5, number = 10, subsets = 2^(2:4))

- *dataset*: a specmine dataset;
- *column.class*: string representing the metadata variable to predict;
- *method*: string representing the feature selection method to use. It can either be Recursive Feature Elimination ("rfe") or Selection by filter ("filter"). Defaults to "rfe";
- *functions*: function to use in model fitting prediction and variable importance/filtering. It can eitherbe:
  - Random Forests: `functions=caret::rfFuncs`, if `method="rfe"`, or `functions=caret::rfSBF`, if `method="filter"`;
  - Linear Regression: `functions=caret::lmFuncs`, if `method="rfe"`, or `functions=caret::lmSBF`, if `method="filter"`;
  - Bagged Trees: `functions=caret::treebagFuncs`, if `method="rfe"`, or `functions=caret::treebagSBF`, if `method="filter"`;
  - Linear Discriminant Analysis (LDA): `functions=caret::ldaFuncs`, if `method="rfe"`, or `functions=caret::ldaSBF`, if `method="filter"`;
  - Naive-Bayes: `functions=caret::nbFuncs`, if `method="rfe"`, or `functions=caret::nbSBF`, if `method="filter"`.
- *validation*: string representing the validation method. It can either be:
  - "boot": Resampling;
  - "cv": Cross-Validation;
  - "repeatedcv": Repeated Cross-validation;
  - "loocv": Leave One Out Cross-Validation;
  - "lgocv": Leave Group Out Cross-Validation.
- *repeats*: if "boot" or "repeatedcv" are chosen, a numeric value with the number of repeats must be given. Defaults to 5;
- *number*: if a cross validation method is chosen, a numeric value with the number of folds must be given. Defaults to 10;
- *subsets*: numeric vector indicating the number of features for each group of test. Only necessary to indicate when method="rfe". Defaults to 2^(2:4).

12.2 Example

Example makes use of an concentrations dataset (concentrations dataset used).

1. Recursive Feature Elimination, where the function chosen to do model fitting prediction and variable importance/filtering is `rfFuncs` (random forests). For model validation, the cross validation method with 3 folds was used:

Hide

```
res_fs_rfe_concentrations=feature_selection(concentrations_dataset, "Muscle.loss", method="rfe", functions = caret::rfFuncs, validatio
n = "cv", number = 3, subsets = 2^(1:6))
```

2. Summary of the results:

Hide

```
res_fs_rfe_concentrations
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (3 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy Kappa AccuracySD KappaSD Selected
##          2    0.662 0.281     0.0271  0.0437
##          4    0.649 0.265     0.0715  0.1531
##          8    0.714 0.386     0.0484  0.0820
##         16    0.715 0.395     0.0548  0.1026
##         32    0.714 0.392     0.0484  0.0930
##         63    0.741 0.450     0.0788  0.1545        *
##
## The top 5 variables (out of 63):
##    X3.Hydroxyisovalerate, Creatine, Methylamine, myo.Inositol, Glucose
```

3. Name of the variables that make the best group:

Hide

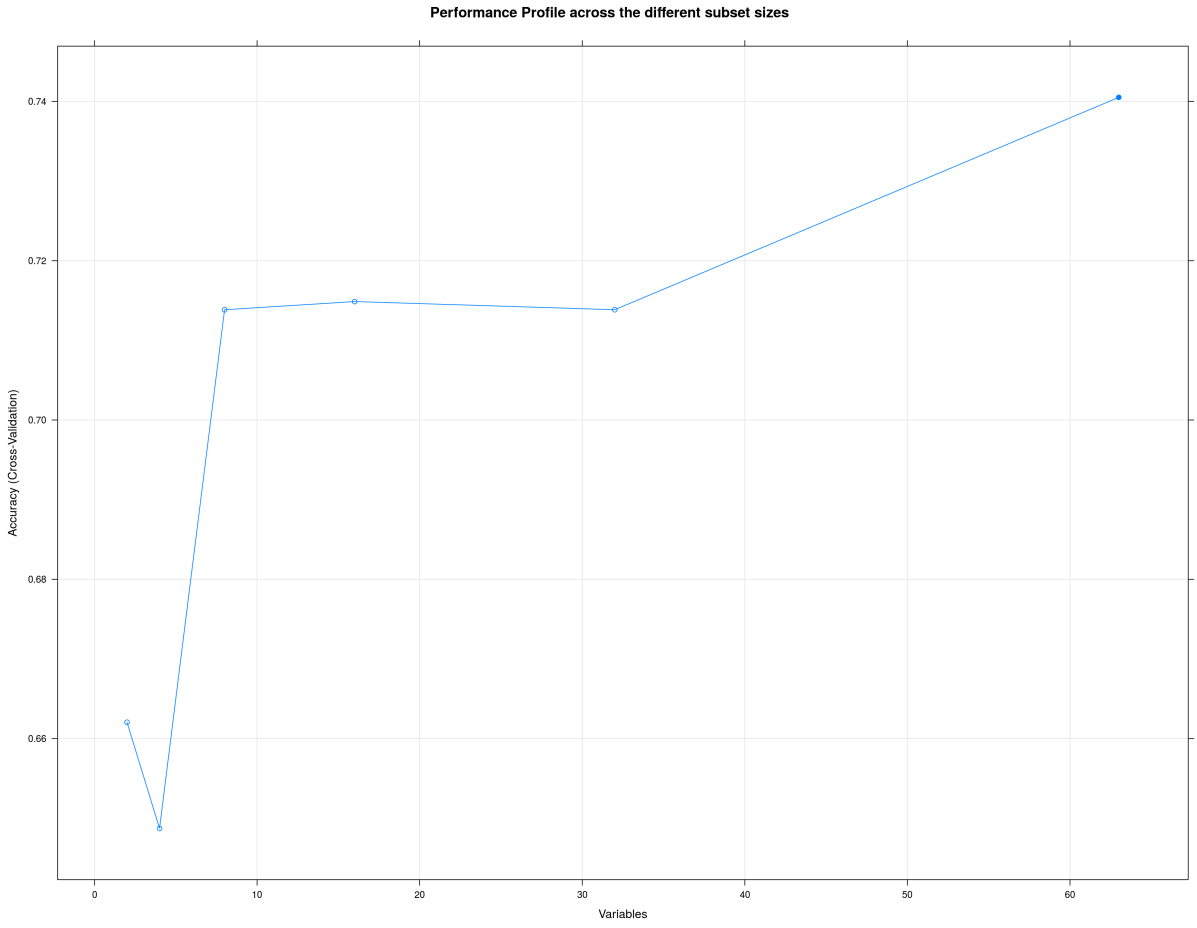
```
res_fs_rfe_concentrations$optVariables
```

```
##  [1] "X3.Hydroxyisovalerate"       "Creatine"                    "Methylamine"                 "myo.Inositol"
##  [5] "Glucose"                     "Adipate"                     "Quinolinate"                 "Glutamine"
##  [9] "Pyroglutamate"               "N.N.Dimethylglycine"         "Sucrose"                     "Histidine"
## [13] "Trimethylamine.N.oxide"      "Formate"                     "Tryptophan"                  "Betaine"
## [17] "Leucine"                     "Acetate"                     "Carnitine"                   "Succinate"
## [21] "Creatinine"                  "Lysine"                      "Xylose"                      "Valine"
## [25] "Asparagine"                  "Threonine"                   "Lactate"                     "Hippurate"
## [29] "Serine"                      "Dimethylamine"               "cis.Aconitate"               "X2.Aminobutyrate"
## [33] "X3.Indoxylsulfate"           "pi.Methylhistidine"          "tau.Methylhistidine"         "Pyruvate"
## [37] "Guanidoacetate"              "O.Acetylcarnitine"           "X3.Aminoisobutyrate"         "X3.Hydroxybutyrate"
## [41] "Alanine"                     "Fumarate"                    "X1.6.Anhydro.beta.D.glucose" "Acetone"
## [45] "Tyrosine"                    "X2.Oxoglutarate"             "Isoleucine"                  "Trigonelline"
## [49] "Ethanolamine"                "Fucose"                      "Citrate"                     "Uracil"
## [53] "Tartrate"                    "Methylguanidine"             "Glycolate"                   "X2.Hydroxyisobutyrate"
## [57] "trans.Aconitate"             "X4.Hydroxyphenylacetate"     "Hypoxanthine"                "Pantothenate"
## [61] "X1.Methylnicotinamide"       "Taurine"                     "Glycine"
```

4. Performance plot:

Hide

```
plot(res_fs_rfe_concentrations, type=c("g","o"), main="Performance Profile across the different subset sizes")
```



13 Regression Analysis

13.1 Functions to use

13.1.1 To run analysis

You can perform linear regression in one specific data variable

> linregression_onevar(dataset, x.val, metadata.vars, combination)

- *dataset*: a specmine dataset;
- *x.val*: string representing the variable to be tested in the linear regression analysis;
- *metadata.vars*: string or character vector indicating the metadata variable(s) to use in the linear regression;
- *combination*: string representing a formula specifying the model. For example, if *metadata.vars* is c("var1", "var2"), combination could be "var1+var2".

You can also perform linear regression in each data variable at once

> linreg_all_vars(dataset, metadata.vars, combination)

- *dataset*: a specmine dataset;
- *metadata.vars*: string or character vector indicating the metadata variable(s) to use in the linear regression;
- *combination*: string representing a formula specifying the model. For example, if *metadata.vars* is c("var1", "var2"), combination could be "var1+var2".

13.1.2 To visualize results

These functions only can be used for results from the function linreg_all_vars.

You can see a dataframe with the coefficient values

> linreg_coef_table(linreg.results, write.file = F, file.out = "linreg-coefs.csv")

- *linreg.results*: variable containing the linear regression results, obtained from linreg_all_vars function;
- *write.file*: boolean value (TRUE or FALSE) indicating if you want the results to be written in a file. Defaults to FALSE;
- *file.out*: string with the file path to write to. You only need to give this information if you have chosen to write the results to a file (write.file=T). Defaults to "linreg-coefs.csv".

You can see a dataframe with the p-values

> linreg_pvalue_table(linreg.results, write.file = F, file.out = "linreg-pvalues.csv")

- *linreg.results*: variable containing the linear regression results, obtained from linreg_all_vars function;
- *write.file*: boolean value (TRUE or FALSE) indicating if you want the results to be written in a file. Defaults to FALSE;
- *file.out*: string with the file path to write to. You only need to give this information if you have chosen to write the results to a file (write.file=T). Defaults to "linreg-pvalues.csv".

You can see a dataframe with the r-squared values

> linreg_rsquared(linreg.results, write.file = F, file.out = "linreg-rsquared.csv")

- *linreg.results*: variable containing the linear regression results, obtained from linreg_all_vars function;
- *write.file*: boolean value (TRUE or FALSE) indicating if you want the results to be written in a file. Defaults to FALSE;
- *file.out*: string with the file path to write to. You only need to give this information if you have chosen to write the results to a file (write.file=T). Defaults to "linreg-rsquared.csv".

You can see a plot of the coefficient and p-values

> plot_regression_coefs_pvalues(linreg.results, bar.col = NULL, coef.size = 5, ...)

- *linreg.results*: variable containing the linear regression results, obtained from linreg_all_vars function;
- *bar.col*: character vector representing the colors of the bars in the plot. One color for each variable represented; **optional**
- *coef.size*: numeric value indicating the relative size of the font of coefficient values;
- ...: additional parameters for function geom_bar.

13.2 Examples

Example makes use of an nmr-peaks dataset (NMR peaks dataset used), already treated for missing values in a previous example.

1. Linear regression on all variables:

Hide

```
res_linreg_nmr_peaks=linreg_all_vars(nmr_peaks_dataset_mv, c("agroregions", "seasons"), "agroregions*seasons")
```

2. Table with the coefficient values for each combination "agroregions*seasons" of two variables:

Hide

```
DT::datatable(linreg_coef_table(res_linreg_nmr_peaks), options=list(scrollX=T))
```

3. Table with the p-values for each combination "agroregions*seasons" of two variables:

Hide

```
DT::datatable(linreg_pvalue_table(res_linreg_nmr_peaks), options=list(scrollX=T))
```

4. Table with the r-squared values for each combination "agroregions*seasons" of two variables:

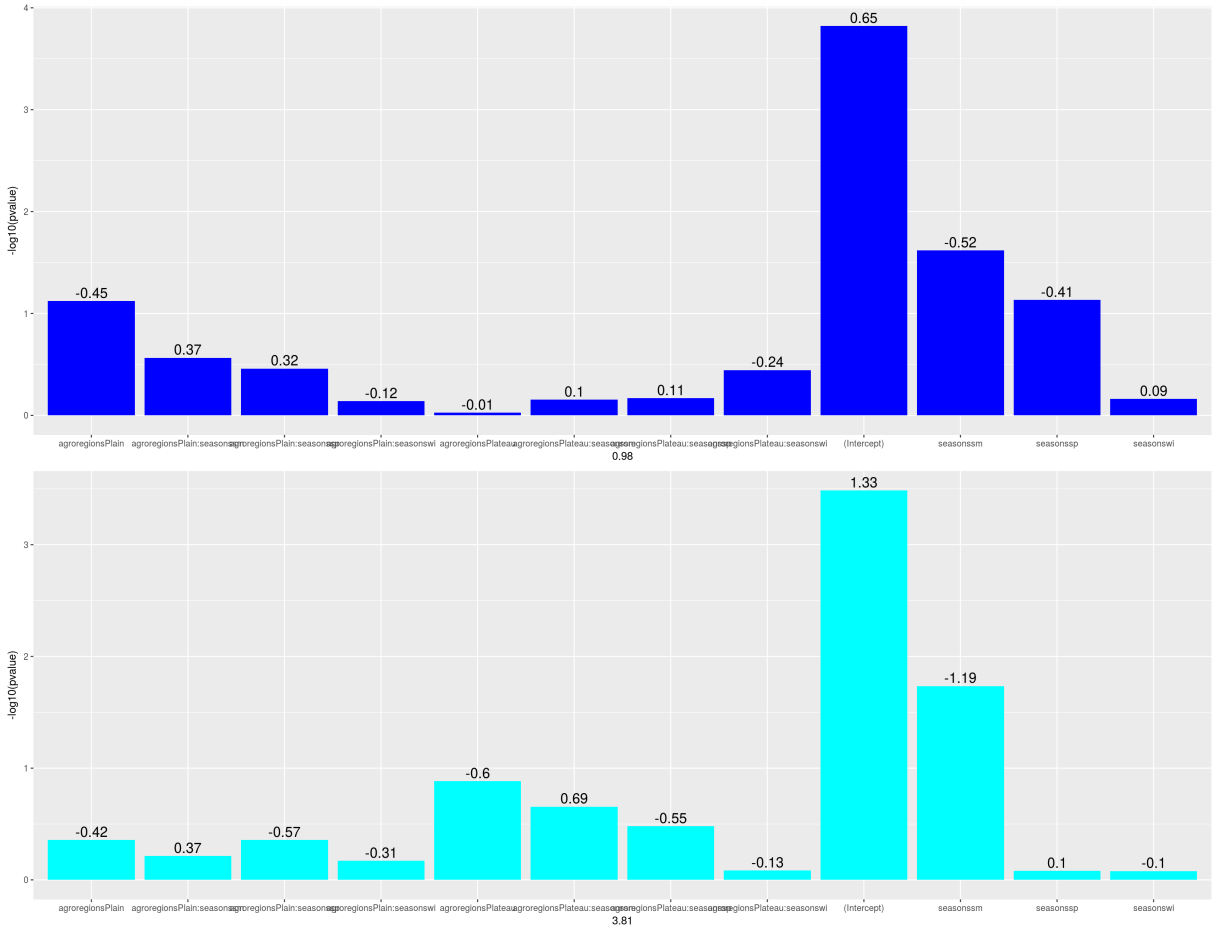
Hide

```
DT::datatable(linreg_rsquared(res_linreg_nmr_peaks), options=list(scrollX=T))
```

5. Plot of the coefficient and p-values of the variables 0.98 and 3.81 in the results:

Hide

```
plot_regression_coefs_pvalues(res_linreg_nmr_peaks[c("0.98", "3.81")], bar.col=c("blue", "cyan"))
```



14 Correlation Analysis

14.1 Functions to use

You can perform a correlations test on two variables or samples from the dataset

> correlation_test(dataset, x, y, method = “pearson”, alternative = “two.sided”, by.var = T)

- *dataset* a specmine dataset;
- *x*: one of the two variables/ samples that will be used in the test;
- *y*: the other variable/ sample that will be used in the test;
- *method*: Correlation method. It can either be “pearson”, “kendall” or “spearman”. Defaults to “pearson”;
- *alternative*: string representing the alternative hypothesis (H1). It can either be:
  - Different from the null hypothesis (H0: x=a and H1: x!=a):“two.sided”;
  - Greater than the null hypothesis (H0: x=a and H1: x>a): “greater”;
  - Less than the null hypothesis (H0: x=a and H1: x<a): “less”;
- *by.var*: boolean value (TRUE or FAlSE) indicating whether the correlations test is being performed to variables (by.var=TRUE) or to samples (by.var=FALSE). Defaults to TRUE.

You can perform correlations test between all variables or samples from the dataset

> correlations_test(dataset, method = “pearson”, by.var = T, alternative = “two.sided”)

- *dataset*: a specmine dataset;
- *method*: Correlation method. It can either be “pearson”, “kendall” or “spearman”. Defaults to “pearson”;
- *by.var*: boolean value (TRUE or FAlSE) indicating whether the correlations test is being performed to variables (by.var=TRUE) or to samples (by.var=FALSE). Defaults to TRUE;
- *alternative*: string representing the alternative hypothesis (H1). It can either be:
  - Different from the null hypothesis (H0: x=a and H1: x!=a):“two.sided”;
  - Greater than the null hypothesis (H0: x=a and H1: x>a): “greater”;
  - Less than the null hypothesis (H0: x=a and H1: x<a): “less”.

You can also calculate the correlations between all variables or samples from the dataset

> correlations_dataset(dataset, method = “pearson”, by.var = T)

- *dataset*: a specmine dataset;
- *method*: Correlation method. It can either be “pearson”, “kendall” or “spearman”. Defaults to “pearson”;
- *by.var*: boolean value (TRUE or FAlSE) indicating whether the correlations test is being performed to variables (by.var=TRUE) or to samples (by.var=FALSE). Defaults to TRUE.

You can plot a heatmap with the correlation values obtained from the previous function

> heatmap_correlations(correlations, col = NULL, ...)

- *correlations*: correlations matrix obtained from function *correlations_dataset*, explained above;
- *col*: character vector with the colors to be used on the heatmap. If not given, predefined color pallette will be used; **optional**
- ...: additional arguments to personalize the heatmap. See function heatmap for details.

14.2 Examples

Example makes use of an concentrations dataset (concentrations dataset used).

1. Perform correlations tests on all samples from the dataset, using Pearson method:

Hide

```
res_cor_test_concentrations = correlations_test(concentrations_dataset, by.var = FALSE)
```

2. Table with the results:

Hide

```
DT::datatable(res_cor_test_concentrations, options=list(scrollX=T))
```

3. Calculate correlations values between all samples from the dataset:

Hide

```
res_cor_val_concentrations=correlations_dataset(concentrations_dataset, by.var=F)
```

4. Heatmap of the correlations calculated in step *3.*:

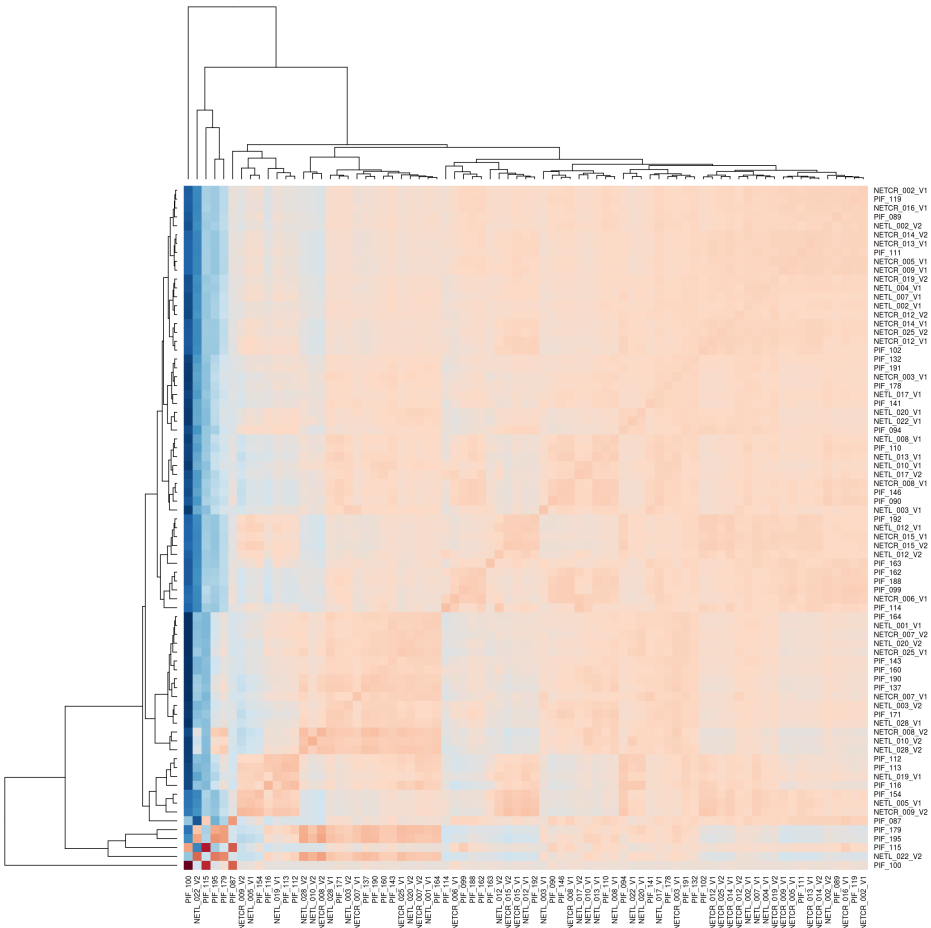
Here is shown an example making use of a color pallette, given through the argument col

See pallette argument from pca_scoresplot2D function for details.

Hide

```
colors=rev(colorRampPalette(RColorBrewer::brewer.pal(10, "RdBu"))(256)) #Obtain a character vector with the color from the 'RdBu' colo
r pallette, in reversed manner. It is this character vector that will be given to the col argument

heatmap_correlations(res_cor_val_concentrations, col=colors)
```



15 Metabolite Identification

Identification of metabolites is only available for LC-MS spectra and NMR Peaks.

15.1 LC-MS Spectra

15.1.1 Function to use

> MAIT_identify_metabolites(dataset, metadata.variable, xSet = NULL, data.folder = NULL, features = NULL, mass.tolerance = 0.5)

- *dataset*: a specmine dataset;
- *metadata.variable*: name of the metadata variable that can help in the identification of the metabolites;
- *xSet*: xcmsSet object that can be passed. Defaults to NULL. Normally, it is xSet=dataset$xSet;
- *data.folder*: string containing the path of the data folder. Should contain the data read into the specmine dataset given;
- *features*: features that can be used to help to identify the metabolites;
- *mass.tolerance*: mass tolerance when matching reference metabolites to the dataset.

15.1.2 Example

Example makes use of this LC-MS dataset.

1. Identification of metabolites. The metadata variable that will help on the identification is "type" (in this dataset, it is the only metadata variable available) and all features (variables, i.e., mass/charge ratios) will be used to identify the metabolites:

Hide

```
library(MAIT)
```

Hide

```
res_id_lcms=MAIT_identify_metabolites(lcms_dataset, "type", features = "all", data.folder=lcms_data_folder, xSet=lcms_dataset$xSet)
```

```
## Warning in dir.create(resultsPath): 'Results_MAIT' already exists
```

```
## WARNING: No input adduct/fragment table was given. Selecting default MAIT table for positive polarity...
## Set adductTable equal to negAdducts to use the default MAIT table for negative polarity
## Start grouping after retention time.
## Created 134 pseudospectra.
## Spectrum build after retention time done
## Generating peak matrix!
## Run isotope peak annotation
##  % finished: 10  20  30  40  50  60  70  80  90  100
## Found isotopes: 65
## Isotope annotation done
## Start grouping after correlation.
## Generating EIC's ..
##
## Calculating peak correlations in 134 Groups...
##  % finished: 10  20  30  40  50  60  70  80  90  100
##
## Calculating peak correlations across samples.
##  % finished: 10  20  30  40  50  60  70  80  90  100
##
## Calculating isotope assignments in 134 Groups...
##  % finished: 10  20  30  40  50  60  70  80  90  100
## Calculating graph cross linking in 134 Groups...
##  % finished: 10  20  30  40  50  60  70  80  90  100
## New number of ps-groups:  179
## xsAnnotate has now 179 groups, instead of 134
## Spectrum number increased after correlation done
## Generating peak matrix for peak annotation!
## Found and use user-defined ruleset!
## Calculating possible adducts in 179 Groups...
##  % finished: 10  20  30  40  50  60  70  80  90  100
## Adduct/fragment annotation done
##
```

```
## Warning in MAIT::peakAnnotation(MAIT.object = mait.object, corrWithSamp = 0.7, : Warning: Folder Results_MAIT/Tables already exists
. Possible file
## overwritting.
```

```
## Skipping peak aggregation step...
```

```
## Warning in MAIT::Biotransformations(MAIT.object = mait.sig, adductAnnotation = T, : No input biotransformations table was given. Se
lecting default
## MAIT table for biotransformations...
```

```
## Warning in MAIT::Biotransformations(MAIT.object = mait.sig, adductAnnotation = T, : No input adduct/fragment table was given. Selec
ting default MAIT
## table for positive polarity...
```

```
## Set adductTable equal to negAdducts to use the default MAIT table for negative polarity
##
##  % Annotation in progress: 10  20  30  40  50  60  70  80  90  100  WARNING: No input database table was given. Selecting default M
AIT database...
## Metabolite identification initiated
##
##  % Metabolite identification in progress: 0  10  20  30  40  50  60  70  80  90  100
##  Metabolite identification finished
```

```
## Warning in MAIT::identifyMetabolites(MAIT.object = mait.sig, peakTolerance = 0.005): Folder /home/scardoso/Documents/specmine/repor
ts/Results_MAIT/
## Tables already exists. Possible file overwritting.
```

2. Table with the results:

Hide

```
DT::datatable(res_id_lcms@FeatureInfo@metaboliteTable, options=list(scrollX=T))
```

3. Table with filtered results. Only known metabolites are here shown, as well only specific columns were selected:

Hide

```
DT::datatable(res_id_lcms@FeatureInfo@metaboliteTable[which(res_id_lcms@FeatureInfo@metaboliteTable$Name!='Unknown'),c(6,9,1,2,3,4,5,7
,8,10,11)], options=list(scrollX=T))
```

15.2 NMR Peaks

15.2.1 Function to use

> nmr_identification(dataset, ppm.tol=0.03, clust.method='pearson', clust.treshold=NULL, clust.peaks.min=2, clust.maxPeaks=40, clust.nTop=5, freq=500, nucl="1H", solv=NULL, pH=NULL, temp=NULL)

- *dataset*: a specmine dataset;
- *ppm.tol*: ppm tolerance when matching reference peaks to the dataset peaks;
- *clust.method*: correlation method to use in the formation of clusters. Defaults to "pearson";
- *clust.treshold*: minimum correlation between variables to form clusters. If not given (clust.treshold=NULL), the function calculates the optimum value (value that leads to the greater number of clusters);
- *clust.peaks.min*: minimum number of variables in each cluster. Only the clusters with at least *clust.peaks.min* variables will be considered.
- *clust.nTop*: number of top metabolites with greater score to show for each cluster;
- *clust.maxPeaks*: maximum number of peaks that a cluster can have, while searching for the best correlation value. Defaults to 40, the original value where the code was adapated from. Can also be NULL and this value will be the number of peaks of the larger cluster.
- *freq*: frequency of reference spectra, in Hz. Must either be 400, 500 or 600.
- *nucl*: atomic nuclei of reference spectra, either "1H" or "13C".
- *solv*: solvent of the sample used to ontain the reference spectra. Possbile values: "100%_DMSO","5%_DMSO","acetone+DMSO+tetramethylurea","acetone+DMSO+tetramethylurea","C","CCl4","CD3OD","CDCl3","cyclohexane","D2O","DMSO-d6","DMSO-d6+HCl","neat","TMS","Water". **optional**
- *pH*: pH of the sample used to ontain the reference spectra. Can be a number corresponding to the sample's pH or a vector of length two indicating a pH interval. **optional**
- *temp*: temperature of the sample used to ontain the reference spectra, in Celsius. Must either be 25 or 50. **optional**

15.2.2 Example

Example makes use of this NMR peaks dataset.

1. Firstly, to perform metabolite identification, the following pre-processing tasks were applied to the dataset, saving the processed data in a new dataset (nmr_peaks_dataset_processed_ID)

- Treatment of missing values, as the dataset contains missing values, by replacing them with the value $5*10^{-4}$:

Hide

```
count_missing_values(nmr_peaks_dataset)
```

```
## [1] 5691
```

Hide

```
nmr_peaks_dataset_processed_ID=missingvalues_imputation(nmr_peaks_dataset)
```

- Logaritmic transformation:

Hide

```
nmr_peaks_dataset_processed_ID=transform_data(nmr_peaks_dataset_processed_ID)
```

- Auto scale:

Hide

```
nmr_peaks_dataset_processed_ID=scaling(nmr_peaks_dataset_processed_ID)
```

2. Now, metabolites will be identified

Hide

```
nmr_ID_res = nmr_identification(nmr_peaks_dataset_processed_ID, clust.maxPeaks = NULL, clust.nTop = 10)
```

```
## Getting reference metabolites...
## Getting best correlation value...
## CVAL;nb_clusters;nb_clusters_2;size_max;Criterion;nb_buckets
## 0.9;13;9;76;-15.34;169
## 0.9001;13;9;76;-15.34;169
## 0.9002;13;9;76;-15.34;169
## 0.9003;13;9;75;-15.22;168
## 0.9004;13;9;75;-15.22;168
## 0.9005;13;9;75;-15.22;168
## 0.9006;13;9;75;-15.22;168
## 0.9007;13;9;75;-15.22;168
## 0.9008;13;9;75;-15.22;168
## 0.9009;13;9;75;-15.22;168
## 0.901;13;9;75;-15.22;168
## 0.9011;13;9;75;-15.22;168
## 0.9012;12;8;75;-15.92;166
## 0.9013;12;8;75;-15.92;166
## 0.9014;12;8;75;-15.92;166
## 0.9015;12;8;75;-15.92;166
## 0.9016;12;8;75;-15.92;166
## 0.9017;12;8;75;-15.92;166
## 0.9018;12;8;75;-15.92;165
## 0.9019;12;8;74;-15.8;164
## 0.902;12;8;74;-15.8;164
## 0.9021;12;8;74;-15.8;164
## 0.9022;12;8;74;-15.8;164
## 0.9023;11;7;74;-16.56;162
## 0.9024;11;7;74;-16.56;162
## 0.9025;11;7;74;-16.56;162
## 0.9026;11;7;74;-16.56;161
## 0.9027;11;7;74;-16.56;161
## 0.9028;11;7;73;-16.44;160
## 0.9029;11;7;72;-16.32;159
## 0.903;11;7;72;-16.32;159
## 0.9031;11;7;72;-16.32;159
## 0.9032;11;7;72;-16.32;159
## 0.9033;11;7;72;-16.32;159
## 0.9034;11;7;72;-16.32;159
## 0.9035;11;7;72;-16.32;159
## 0.9036;11;7;72;-16.32;159
## 0.9037;11;7;72;-16.32;158
## 0.9038;11;7;72;-16.32;158
## 0.9039;12;7;72;-15.56;158
## 0.904;12;7;72;-15.56;158
## 0.9041;12;7;72;-15.56;158
## 0.9042;13;7;72;-14.87;158
## 0.9043;13;7;72;-14.87;158
## 0.9044;13;7;72;-14.87;158
## 0.9045;13;7;72;-14.87;158
## 0.9046;13;7;72;-14.87;158
## 0.9047;13;7;72;-14.87;158
## 0.9048;13;7;72;-14.87;158
## 0.9049;13;7;72;-14.87;158
## 0.905;13;7;72;-14.87;158
## 0.9051;13;7;72;-14.87;157
## 0.9052;13;7;72;-14.87;156
## 0.9053;13;7;72;-14.87;156
## 0.9054;13;7;72;-14.87;156
## 0.9055;14;8;72;-14.22;156
## 0.9056;14;8;72;-14.22;156
## 0.9057;14;8;72;-14.22;156
## 0.9058;14;8;72;-14.22;156
## 0.9059;14;8;72;-14.22;156
## 0.906;14;8;72;-14.22;156
## 0.9061;14;8;72;-14.22;156
## 0.9062;14;8;72;-14.22;156
## 0.9063;14;8;72;-14.22;156
## 0.9064;14;8;72;-14.22;156
## 0.9065;14;8;72;-14.22;156
## 0.9066;14;8;72;-14.22;156
## 0.9067;14;8;72;-14.22;156
## 0.9068;14;8;72;-14.22;155
## 0.9069;14;8;72;-14.22;155
## 0.907;14;8;72;-14.22;155
## 0.9071;14;8;72;-14.22;155
## 0.9072;14;8;72;-14.22;155
## 0.9073;14;8;72;-14.22;155
## 0.9074;14;8;72;-14.22;154
## 0.9075;14;8;72;-14.22;154
## 0.9076;14;8;72;-14.22;154
## 0.9077;14;8;72;-14.22;154
## 0.9078;14;8;72;-14.22;154
## 0.9079;14;8;72;-14.22;154
## 0.908;14;8;72;-14.22;154
## 0.9081;14;8;71;-14.1;153
## 0.9082;14;8;71;-14.1;153
## 0.9083;14;8;71;-14.1;153
## 0.9084;14;8;71;-14.1;153
## 0.9085;14;8;71;-14.1;153
## 0.9086;14;8;71;-14.1;153
## 0.9087;14;8;71;-14.1;153
## 0.9088;14;8;71;-14.1;153
```

```
## 0.9089;14;8;71;-14.1;153
## 0.909;14;8;71;-14.1;153
## 0.9091;15;8;44;-9.347;153
## 0.9092;15;8;44;-9.347;152
## 0.9093;15;8;44;-9.347;152
## 0.9094;15;8;44;-9.347;151
## 0.9095;15;8;44;-9.347;151
## 0.9096;15;8;44;-9.347;151
## 0.9097;15;8;44;-9.347;151
## 0.9098;16;9;44;-8.787;151
## 0.9099;16;9;44;-8.787;151
## 0.91;16;9;44;-8.787;151
## 0.9101;16;9;44;-8.787;151
## 0.9102;16;9;44;-8.787;150
## 0.9103;16;9;44;-8.787;150
## 0.9104;16;9;44;-8.787;149
## 0.9105;16;9;44;-8.787;149
## 0.9106;16;9;44;-8.787;149
## 0.9107;15;8;44;-9.347;147
## 0.9108;15;8;44;-9.347;147
## 0.9109;15;8;44;-9.347;147
## 0.911;14;7;44;-9.946;145
## 0.9111;14;7;44;-9.946;145
## 0.9112;14;7;44;-9.946;145
## 0.9113;14;7;44;-9.946;145
## 0.9114;14;7;44;-9.946;145
## 0.9115;14;7;44;-9.946;145
## 0.9116;14;7;44;-9.946;145
## 0.9117;14;7;44;-9.946;145
## 0.9118;14;7;44;-9.946;145
## 0.9119;14;7;44;-9.946;145
## 0.912;14;7;44;-9.946;145
## 0.9121;14;7;44;-9.946;145
## 0.9122;15;8;44;-9.347;145
## 0.9123;15;8;44;-9.347;144
## 0.9124;15;8;44;-9.347;144
## 0.9125;15;8;44;-9.347;144
## 0.9126;15;8;44;-9.347;144
## 0.9127;15;8;44;-9.347;144
## 0.9128;15;8;44;-9.347;144
## 0.9129;15;8;44;-9.347;144
## 0.913;15;8;44;-9.347;144
## 0.9131;15;8;44;-9.347;144
## 0.9132;15;8;44;-9.347;144
## 0.9133;15;8;44;-9.347;144
## 0.9134;14;7;44;-9.946;142
## 0.9135;14;7;44;-9.946;142
## 0.9136;14;7;44;-9.946;142
## 0.9137;14;7;43;-9.747;141
## 0.9138;14;7;43;-9.747;141
## 0.9139;14;7;43;-9.747;141
## 0.914;14;7;43;-9.747;141
## 0.9141;14;7;43;-9.747;141
## 0.9142;14;7;43;-9.747;140
## 0.9143;14;7;43;-9.747;140
## 0.9144;14;7;43;-9.747;140
## 0.9145;14;7;43;-9.747;140
## 0.9146;14;7;43;-9.747;140
## 0.9147;14;7;43;-9.747;140
## 0.9148;14;7;43;-9.747;139
## 0.9149;14;7;43;-9.747;139
## 0.915;13;6;43;-10.39;137
## 0.9151;13;6;43;-10.39;137
## 0.9152;13;6;43;-10.39;137
## 0.9153;13;6;43;-10.39;137
## 0.9154;14;6;43;-9.747;137
## 0.9155;14;6;43;-9.747;137
## 0.9156;14;6;43;-9.747;137
## 0.9157;14;6;42;-9.542;136
## 0.9158;14;6;42;-9.542;136
## 0.9159;13;5;42;-10.19;134
## 0.916;13;5;42;-10.19;133
## 0.9161;13;5;42;-10.19;133
## 0.9162;13;5;42;-10.19;133
## 0.9163;13;5;42;-10.19;133
## 0.9164;13;5;42;-10.19;133
## 0.9165;13;5;42;-10.19;133
## 0.9166;13;5;42;-10.19;133
## 0.9167;13;5;42;-10.19;133
## 0.9168;13;5;42;-10.19;133
## 0.9169;13;5;42;-10.19;133
## 0.917;13;5;42;-10.19;133
## 0.9171;13;5;42;-10.19;133
## 0.9172;13;5;42;-10.19;133
## 0.9173;13;5;42;-10.19;133
## 0.9174;13;5;42;-10.19;133
## 0.9175;13;5;42;-10.19;133
## 0.9176;13;5;42;-10.19;132
## 0.9177;13;5;42;-10.19;132
## 0.9178;13;5;42;-10.19;132
## 0.9179;13;5;42;-10.19;132
## 0.918;13;5;42;-10.19;132
## 0.9181;13;5;42;-10.19;132
## 0.9182;13;5;42;-10.19;132
## 0.9183;13;5;42;-10.19;132
```

```
## 0.9184;13;5;42;-10.19;132
## 0.9185;13;5;42;-10.19;132
## 0.9186;13;5;42;-10.19;132
## 0.9187;13;5;42;-10.19;132
## 0.9188;13;5;42;-10.19;132
## 0.9189;13;5;42;-10.19;132
## 0.919;13;5;42;-10.19;132
## 0.9191;13;5;42;-10.19;132
## 0.9192;14;6;42;-9.542;132
## 0.9193;14;6;42;-9.542;132
## 0.9194;14;6;42;-9.542;132
## 0.9195;14;6;42;-9.542;132
## 0.9196;14;6;42;-9.542;132
## 0.9197;14;6;42;-9.542;132
## 0.9198;14;6;42;-9.542;132
## 0.9199;14;6;42;-9.542;132
## 0.92;14;6;42;-9.542;132
## 0.9201;14;6;42;-9.542;132
## 0.9202;14;6;42;-9.542;132
## 0.9203;14;6;42;-9.542;132
## 0.9204;14;6;42;-9.542;132
## 0.9205;14;6;42;-9.542;132
## 0.9206;14;6;42;-9.542;132
## 0.9207;14;6;42;-9.542;132
## 0.9208;14;6;42;-9.542;132
## 0.9209;14;6;42;-9.542;132
## 0.921;14;6;42;-9.542;132
## 0.9211;14;6;42;-9.542;132
## 0.9212;14;6;42;-9.542;132
## 0.9213;14;6;41;-9.333;131
## 0.9214;14;6;41;-9.333;131
## 0.9215;14;6;41;-9.333;131
## 0.9216;13;5;41;-9.977;129
## 0.9217;13;5;41;-9.977;129
## 0.9218;13;5;41;-9.977;129
## 0.9219;14;5;38;-8.673;129
## 0.922;14;5;38;-8.673;129
## 0.9221;14;5;38;-8.673;129
## 0.9222;14;5;38;-8.673;128
## 0.9223;14;5;38;-8.673;128
## 0.9224;14;5;38;-8.673;128
## 0.9225;14;5;38;-8.673;128
## 0.9226;14;5;38;-8.673;128
## 0.9227;14;5;38;-8.673;128
## 0.9228;14;5;37;-8.441;127
## 0.9229;14;5;37;-8.441;127
## 0.923;14;5;37;-8.441;127
## 0.9231;14;5;37;-8.441;127
## 0.9232;14;5;37;-8.441;127
## 0.9233;14;5;37;-8.441;127
## 0.9234;14;5;37;-8.441;127
## 0.9235;14;5;37;-8.441;127
## 0.9236;14;5;37;-8.441;127
## 0.9237;14;5;37;-8.441;127
## 0.9238;14;5;37;-8.441;127
## 0.9239;14;5;37;-8.441;127
## 0.924;14;5;37;-8.441;127
## 0.9241;14;5;37;-8.441;127
## 0.9242;14;5;37;-8.441;127
## 0.9243;14;5;37;-8.441;127
## 0.9244;14;5;37;-8.441;127
## 0.9245;14;5;37;-8.441;127
## 0.9246;14;5;37;-8.441;127
## 0.9247;14;5;37;-8.441;127
## 0.9248;14;5;36;-8.203;126
## 0.9249;14;5;36;-8.203;126
## 0.925;14;5;36;-8.203;126
## 0.9251;14;5;36;-8.203;126
## 0.9252;14;5;36;-8.203;125
## 0.9253;14;5;36;-8.203;125
## 0.9254;14;5;36;-8.203;125
## 0.9255;14;5;36;-8.203;125
## 0.9256;14;5;36;-8.203;125
## 0.9257;14;5;36;-8.203;125
## 0.9258;14;5;36;-8.203;125
## 0.9259;14;5;36;-8.203;125
## 0.926;14;5;36;-8.203;125
## 0.9261;14;5;36;-8.203;125
## 0.9262;14;5;36;-8.203;125
## 0.9263;14;5;36;-8.203;125
## 0.9264;14;5;36;-8.203;125
## 0.9265;14;5;36;-8.203;125
## 0.9266;14;5;36;-8.203;125
## 0.9267;14;6;36;-8.203;124
## 0.9268;14;6;36;-8.203;124
## 0.9269;14;6;36;-8.203;124
## 0.927;14;6;36;-8.203;124
## 0.9271;14;6;36;-8.203;124
## 0.9272;14;6;36;-8.203;124
## 0.9273;14;6;36;-8.203;124
## 0.9274;13;5;36;-8.847;122
## 0.9275;13;5;36;-8.847;122
## 0.9276;13;5;36;-8.847;122
## 0.9277;13;5;36;-8.847;122
## 0.9278;13;5;36;-8.847;122
```

```
## 0.9279;13;5;36;-8.847;122
## 0.928;13;5;36;-8.847;122
## 0.9281;13;5;36;-8.847;122
## 0.9282;13;5;36;-8.847;122
## 0.9283;13;5;36;-8.847;122
## 0.9284;13;5;36;-8.847;122
## 0.9285;13;5;36;-8.847;122
## 0.9286;13;5;36;-8.847;122
## 0.9287;13;5;36;-8.847;122
## 0.9288;13;5;36;-8.847;122
## 0.9289;13;5;36;-8.847;122
## 0.929;13;5;36;-8.847;122
## 0.9291;13;5;36;-8.847;122
## 0.9292;13;5;36;-8.847;122
## 0.9293;13;5;36;-8.847;122
## 0.9294;13;5;36;-8.847;122
## 0.9295;13;5;36;-8.847;122
## 0.9296;13;5;36;-8.847;122
## 0.9297;13;5;36;-8.847;122
## 0.9298;13;5;36;-8.847;122
## 0.9299;13;5;36;-8.847;122
## 0.93;13;5;36;-8.847;122
## 0.9301;13;5;36;-8.847;122
## 0.9302;13;5;36;-8.847;122
## 0.9303;13;5;36;-8.847;122
## 0.9304;13;5;36;-8.847;122
## 0.9305;13;5;36;-8.847;122
## 0.9306;13;5;36;-8.847;122
## 0.9307;13;5;36;-8.847;122
## 0.9308;12;4;36;-9.542;120
## 0.9309;12;4;36;-9.542;120
## 0.931;12;4;36;-9.542;120
## 0.9311;12;4;36;-9.542;120
## 0.9312;12;4;36;-9.542;120
## 0.9313;12;4;36;-9.542;120
## 0.9314;12;4;36;-9.542;120
## 0.9315;12;4;36;-9.542;120
## 0.9316;12;4;36;-9.542;120
## 0.9317;12;4;36;-9.542;120
## 0.9318;12;4;36;-9.542;120
## 0.9319;12;4;36;-9.542;120
## 0.932;12;4;36;-9.542;120
## 0.9321;12;4;36;-9.542;120
## 0.9322;12;4;36;-9.542;120
## 0.9323;12;4;36;-9.542;120
## 0.9324;12;4;36;-9.542;120
## 0.9325;12;4;36;-9.542;120
## 0.9326;11;3;36;-10.3;118
## 0.9327;11;3;36;-10.3;118
## 0.9328;11;3;36;-10.3;118
## 0.9329;11;3;36;-10.3;118
## 0.933;11;3;36;-10.3;118
## 0.9331;11;3;36;-10.3;118
## 0.9332;11;3;36;-10.3;118
## 0.9333;11;3;36;-10.3;118
## 0.9334;11;3;36;-10.3;118
## 0.9335;11;3;36;-10.3;118
## 0.9336;11;3;36;-10.3;118
## 0.9337;11;3;36;-10.3;118
## 0.9338;11;3;36;-10.3;118
## 0.9339;12;4;36;-9.542;118
## 0.934;12;4;36;-9.542;118
## 0.9341;12;4;36;-9.542;118
## 0.9342;14;4;31;-6.905;118
## 0.9343;14;4;31;-6.905;118
## 0.9344;14;4;31;-6.905;118
## 0.9345;14;4;31;-6.905;118
## 0.9346;14;4;31;-6.905;118
## 0.9347;14;4;31;-6.905;117
## 0.9348;14;4;31;-6.905;117
## 0.9349;14;4;31;-6.905;117
## 0.935;14;4;31;-6.905;117
## 0.9351;14;4;31;-6.905;117
## 0.9352;14;4;31;-6.905;117
## 0.9353;14;4;31;-6.905;117
## 0.9354;14;4;31;-6.905;117
## 0.9355;14;4;31;-6.905;117
## 0.9356;14;4;31;-6.905;117
## 0.9357;14;4;31;-6.905;117
## 0.9358;14;4;31;-6.905;117
## 0.9359;14;4;31;-6.905;117
## 0.936;14;4;31;-6.905;117
## 0.9361;14;4;31;-6.905;117
## 0.9362;14;4;31;-6.905;117
## 0.9363;14;4;31;-6.905;117
## 0.9364;14;4;31;-6.905;117
## 0.9365;14;4;31;-6.905;117
## 0.9366;14;4;31;-6.905;117
## 0.9367;14;4;31;-6.905;117
## 0.9368;15;4;31;-6.305;117
## 0.9369;15;4;31;-6.305;117
## 0.937;15;4;31;-6.305;116
## 0.9371;15;4;31;-6.305;116
## 0.9372;15;4;31;-6.305;116
```

```
## 0.9373;15;4;31;-6.305;116
## 0.9374;16;4;31;-5.745;116
## 0.9375;16;4;31;-5.745;115
## 0.9376;16;4;31;-5.745;114
## 0.9377;16;4;31;-5.745;114
## 0.9378;16;4;31;-5.745;114
## 0.9379;16;4;30;-5.46;113
## 0.938;16;4;30;-5.46;113
## 0.9381;16;4;30;-5.46;113
## 0.9382;16;4;30;-5.46;112
## 0.9383;16;4;30;-5.46;112
## 0.9384;16;4;30;-5.46;112
## 0.9385;16;4;30;-5.46;112
## 0.9386;16;4;30;-5.46;112
## 0.9387;16;4;30;-5.46;112
## 0.9388;16;4;30;-5.46;112
## 0.9389;16;4;30;-5.46;112
## 0.939;16;4;30;-5.46;112
## 0.9391;16;4;30;-5.46;112
## 0.9392;16;4;30;-5.46;112
## 0.9393;16;4;30;-5.46;112
## 0.9394;16;4;30;-5.46;112
## 0.9395;16;4;30;-5.46;112
## 0.9396;16;4;30;-5.46;112
## 0.9397;16;4;30;-5.46;112
## 0.9398;16;4;29;-5.166;111
## 0.9399;16;4;29;-5.166;111
## 0.94;16;4;29;-5.166;111
## 0.9401;16;4;29;-5.166;111
## 0.9402;16;4;29;-5.166;111
## 0.9403;16;4;29;-5.166;110
## 0.9404;16;4;29;-5.166;110
## 0.9405;16;4;29;-5.166;110
## 0.9406;16;4;29;-5.166;110
## 0.9407;16;4;29;-5.166;110
## 0.9408;16;4;29;-5.166;110
## 0.9409;16;4;29;-5.166;110
## 0.941;16;4;29;-5.166;109
## 0.9411;16;4;29;-5.166;109
## 0.9412;16;4;29;-5.166;108
## 0.9413;16;4;29;-5.166;108
## 0.9414;16;4;29;-5.166;108
## 0.9415;16;4;29;-5.166;108
## 0.9416;16;4;29;-5.166;108
## 0.9417;16;4;29;-5.166;108
## 0.9418;16;4;29;-5.166;108
## 0.9419;17;5;29;-4.639;108
## 0.942;17;5;29;-4.639;108
## 0.9421;17;5;29;-4.639;108
## 0.9422;17;5;29;-4.639;108
## 0.9423;17;5;29;-4.639;108
## 0.9424;17;5;29;-4.639;108
## 0.9425;17;5;29;-4.639;108
## 0.9426;17;5;29;-4.639;108
## 0.9427;17;5;29;-4.639;108
## 0.9428;17;5;29;-4.639;108
## 0.9429;17;5;29;-4.639;108
## 0.943;17;5;29;-4.639;108
## 0.9431;16;4;29;-5.166;106
## 0.9432;16;4;29;-5.166;106
## 0.9433;16;4;29;-5.166;106
## 0.9434;16;4;29;-5.166;106
## 0.9435;16;4;29;-5.166;106
## 0.9436;16;4;29;-5.166;106
## 0.9437;16;4;29;-5.166;106
## 0.9438;16;4;29;-5.166;106
## 0.9439;16;4;29;-5.166;105
## 0.944;16;4;29;-5.166;105
## 0.9441;16;4;29;-5.166;105
## 0.9442;16;4;29;-5.166;105
## 0.9443;16;4;29;-5.166;104
## 0.9444;16;4;29;-5.166;104
## 0.9445;16;4;29;-5.166;104
## 0.9446;16;4;29;-5.166;104
## 0.9447;16;4;29;-5.166;104
## 0.9448;16;4;29;-5.166;104
## 0.9449;16;4;29;-5.166;104
## 0.945;16;4;29;-5.166;104
## 0.9451;16;4;29;-5.166;104
## 0.9452;16;4;29;-5.166;104
## 0.9453;17;5;29;-4.639;104
## 0.9454;17;5;29;-4.639;104
## 0.9455;17;5;29;-4.639;104
## 0.9456;17;5;29;-4.639;104
## 0.9457;17;5;29;-4.639;104
## 0.9458;17;5;29;-4.639;104
## 0.9459;17;5;29;-4.639;104
## 0.946;17;5;29;-4.639;104
## 0.9461;17;5;29;-4.639;104
## 0.9462;17;5;29;-4.639;104
## 0.9463;18;6;29;-4.143;104
## 0.9464;18;6;29;-4.143;104
## 0.9465;18;6;29;-4.143;104
## 0.9466;18;6;29;-4.143;104
## 0.9467;18;7;29;-4.143;103
## 0.9468;18;7;29;-4.143;103
```

```
## 0.9468;18;7;29;-4.143;103
## 0.9469;18;7;29;-4.143;103
## 0.947;18;7;29;-4.143;103
## 0.9471;18;7;29;-4.143;103
## 0.9472;18;7;29;-4.143;103
## 0.9473;18;7;29;-4.143;103
## 0.9474;18;7;29;-4.143;103
## 0.9475;18;7;29;-4.143;103
## 0.9476;18;7;29;-4.143;103
## 0.9477;18;7;29;-4.143;103
## 0.9478;18;7;29;-4.143;103
## 0.9479;18;7;29;-4.143;103
## 0.948;18;7;29;-4.143;103
## 0.9481;18;7;29;-4.143;103
## 0.9482;18;7;29;-4.143;102
## 0.9483;18;7;29;-4.143;102
## 0.9484;18;7;29;-4.143;102
## 0.9485;18;7;29;-4.143;102
## 0.9486;18;7;29;-4.143;102
## 0.9487;18;7;29;-4.143;102
## 0.9488;18;7;29;-4.143;102
## 0.9489;18;7;29;-4.143;102
## 0.949;18;7;29;-4.143;102
## 0.9491;18;7;29;-4.143;102
## 0.9492;18;7;29;-4.143;101
## 0.9493;18;7;29;-4.143;101
## 0.9494;18;8;29;-4.143;100
## 0.9495;18;8;29;-4.143;100
## 0.9496;18;8;29;-4.143;100
## 0.9497;18;8;29;-4.143;100
## 0.9498;18;8;29;-4.143;99
## 0.9499;18;8;29;-4.143;99
## 0.95;19;8;29;-3.673;99
## 0.9501;19;8;29;-3.673;99
## 0.9502;19;8;29;-3.673;99
## 0.9503;20;9;29;-3.227;99
## 0.9504;20;9;29;-3.227;99
## 0.9505;20;9;29;-3.227;99
## 0.9506;20;9;29;-3.227;99
## 0.9507;20;9;29;-3.227;99
## 0.9508;20;9;29;-3.227;99
## 0.9509;20;9;29;-3.227;99
## 0.951;19;8;28;-3.368;96
## 0.9511;19;8;28;-3.368;96
## 0.9512;19;8;28;-3.368;96
## 0.9513;19;8;28;-3.368;96
## 0.9514;19;8;28;-3.368;96
## 0.9515;19;8;28;-3.368;96
## 0.9516;19;8;28;-3.368;96
## 0.9517;19;8;28;-3.368;96
## 0.9518;19;9;28;-3.368;95
## 0.9519;19;9;28;-3.368;95
## 0.952;19;9;28;-3.368;95
## 0.9521;19;9;28;-3.368;95
## 0.9522;19;9;28;-3.368;95
## 0.9523;18;8;28;-3.838;93
## 0.9524;18;8;28;-3.838;93
## 0.9525;18;8;28;-3.838;93
## 0.9526;18;8;28;-3.838;92
## 0.9527;18;8;28;-3.838;92
## 0.9528;18;8;28;-3.838;92
## 0.9529;18;8;28;-3.838;92
## 0.953;18;8;28;-3.838;92
## 0.9531;18;8;28;-3.838;92
## 0.9532;18;8;28;-3.838;92
## 0.9533;18;8;28;-3.838;92
## 0.9534;18;8;28;-3.838;92
## 0.9535;18;8;28;-3.838;92
## 0.9536;18;8;28;-3.838;92
## 0.9537;18;8;28;-3.838;92
## 0.9538;18;8;28;-3.838;92
## 0.9539;18;8;28;-3.838;92
## 0.954;18;8;28;-3.838;92
## 0.9541;18;8;28;-3.838;92
## 0.9542;18;8;28;-3.838;92
## 0.9543;18;8;28;-3.838;92
## 0.9544;18;8;28;-3.838;92
## 0.9545;18;8;28;-3.838;92
## 0.9546;18;8;28;-3.838;92
## 0.9547;18;8;28;-3.838;92
## 0.9548;17;7;28;-4.334;90
## 0.9549;17;7;28;-4.334;90
## 0.955;17;7;28;-4.334;90
## 0.9551;17;7;28;-4.334;90
## 0.9552;17;7;28;-4.334;90
## 0.9553;17;7;28;-4.334;90
## 0.9554;17;7;28;-4.334;90
## 0.9555;17;7;28;-4.334;90
## 0.9556;17;7;28;-4.334;90
## 0.9557;17;7;28;-4.334;90
## 0.9558;17;8;28;-4.334;89
## 0.9559;17;8;28;-4.334;89
## 0.956;17;8;28;-4.334;89
## 0.9561;17;8;28;-4.334;89
## 0.9562;17;8;28;-4.334;89
```

```
## 0.9563;17;8;28;-4.334;89
## 0.9564;17;8;28;-4.334;89
## 0.9565;17;8;28;-4.334;89
## 0.9566;17;9;28;-4.334;88
## 0.9567;17;9;28;-4.334;87
## 0.9568;17;9;28;-4.334;87
## 0.9569;17;9;28;-4.334;87
## 0.957;17;9;28;-4.334;87
## 0.9571;17;9;28;-4.334;87
## 0.9572;17;9;28;-4.334;87
## 0.9573;17;9;28;-4.334;87
## 0.9574;17;9;28;-4.334;86
## 0.9575;17;9;28;-4.334;86
## 0.9576;17;9;28;-4.334;86
## 0.9577;17;9;28;-4.334;86
## 0.9578;17;9;28;-4.334;86
## 0.9579;17;9;28;-4.334;86
## 0.958;17;9;28;-4.334;86
## 0.9581;17;9;28;-4.334;86
## 0.9582;17;9;28;-4.334;86
## 0.9583;17;9;28;-4.334;86
## 0.9584;17;9;28;-4.334;86
## 0.9585;17;9;28;-4.334;86
## 0.9586;17;9;28;-4.334;86
## 0.9587;17;9;28;-4.334;86
## 0.9588;17;9;28;-4.334;86
## 0.9589;17;9;28;-4.334;86
## 0.959;16;8;28;-4.861;84
## 0.9591;16;8;28;-4.861;84
## 0.9592;16;8;28;-4.861;84
## 0.9593;16;8;27;-4.545;83
## 0.9594;16;8;27;-4.545;83
## 0.9595;16;8;27;-4.545;83
## 0.9596;16;8;27;-4.545;83
## 0.9597;16;8;27;-4.545;83
## 0.9598;16;8;26;-4.217;82
## 0.9599;16;8;26;-4.217;82
## 0.96;16;8;26;-4.217;82
## 0.9601;16;8;26;-4.217;82
## 0.9602;16;8;26;-4.217;82
## 0.9603;17;8;21;-1.835;82
## 0.9604;17;8;21;-1.835;82
## 0.9605;17;8;21;-1.835;82
## 0.9606;17;8;21;-1.835;82
## 0.9607;17;8;21;-1.835;82
## 0.9608;17;8;21;-1.835;82
## 0.9609;16;7;21;-2.362;80
## 0.961;17;8;20;-1.412;79
## 0.9611;18;8;11;4.278;78
## 0.9612;18;8;11;4.278;78
## 0.9613;18;8;11;4.278;78
## 0.9614;18;8;11;4.278;78
## 0.9615;18;8;11;4.278;78
## 0.9616;18;8;11;4.278;78
## 0.9617;18;8;11;4.278;77
## 0.9618;18;8;11;4.278;77
## 0.9619;18;8;11;4.278;77
## 0.962;18;8;11;4.278;76
## 0.9621;18;8;11;4.278;76
## 0.9622;18;8;11;4.278;76
## 0.9623;18;8;11;4.278;76
## 0.9624;18;8;11;4.278;76
## 0.9625;18;8;11;4.278;76
## 0.9626;18;8;11;4.278;76
## 0.9627;18;8;11;4.278;76
## 0.9628;17;7;11;3.781;74
## 0.9629;17;7;11;3.781;74
## 0.963;17;7;11;3.781;74
## 0.9631;17;7;11;3.781;74
## 0.9632;17;7;11;3.781;74
## 0.9633;17;7;11;3.781;74
## 0.9634;17;7;11;3.781;74
## 0.9635;17;7;11;3.781;74
## 0.9636;17;7;11;3.781;74
## 0.9637;17;7;11;3.781;74
## 0.9638;17;7;11;3.781;74
## 0.9639;17;7;11;3.781;74
## 0.964;17;7;11;3.781;74
## 0.9641;17;7;11;3.781;74
## 0.9642;17;7;11;3.781;74
## 0.9643;17;7;11;3.781;74
## 0.9644;17;7;11;3.781;74
## 0.9645;17;7;11;3.781;74
## 0.9646;17;7;11;3.781;72
## 0.9647;16;6;11;3.255;70
## 0.9648;16;6;11;3.255;70
## 0.9649;16;6;11;3.255;70
## 0.965;16;6;11;3.255;70
## 0.9651;16;6;11;3.255;70
## 0.9652;16;6;11;3.255;70
## 0.9653;16;6;11;3.255;70
## 0.9654;16;6;11;3.255;70
## 0.9655;16;6;11;3.255;70
## 0.9656;16;6;11;3.255;70
## 0.9657;16;6;11;3.255;70
```

```
## 0.9658;16;6;11;3.255;70
## 0.9659;16;6;11;3.255;70
## 0.966;16;6;11;3.255;70
## 0.9661;16;6;11;3.255;70
## 0.9662;16;6;11;3.255;70
## 0.9663;16;6;11;3.255;70
## 0.9664;16;6;11;3.255;69
## 0.9665;16;6;11;3.255;69
## 0.9666;15;5;11;2.694;67
## 0.9667;15;5;11;2.694;67
## 0.9668;15;5;11;2.694;67
## 0.9669;15;6;11;2.694;66
## 0.967;15;6;11;2.694;66
## 0.9671;15;6;11;2.694;66
## 0.9672;14;5;11;2.095;64
## 0.9673;14;5;11;2.095;64
## 0.9674;14;5;11;2.095;64
## 0.9675;14;5;11;2.095;64
## 0.9676;14;5;11;2.095;64
## 0.9677;14;5;11;2.095;64
## 0.9678;14;5;11;2.095;64
## 0.9679;14;5;11;2.095;63
## 0.968;14;5;11;2.095;63
## 0.9681;14;5;11;2.095;63
## 0.9682;14;5;11;2.095;63
## 0.9683;14;5;11;2.095;63
## 0.9684;14;5;11;2.095;62
## 0.9685;14;5;11;2.095;62
## 0.9686;14;5;11;2.095;62
## 0.9687;14;5;11;2.095;62
## 0.9688;14;5;11;2.095;62
## 0.9689;13;4;11;1.451;60
## 0.969;13;4;11;1.451;60
## 0.9691;13;4;11;1.451;60
## 0.9692;13;4;11;1.451;60
## 0.9693;13;4;11;1.451;60
## 0.9694;13;4;11;1.451;60
## 0.9695;13;4;11;1.451;60
## 0.9696;13;4;11;1.451;60
## 0.9697;13;4;11;1.451;59
## 0.9698;13;4;11;1.451;59
## 0.9699;13;4;11;1.451;59
## 0.97;13;4;11;1.451;59
## 0.9701;13;4;11;1.451;59
## 0.9702;13;4;11;1.451;58
## 0.9703;13;4;11;1.451;58
## 0.9704;13;4;11;1.451;58
## 0.9705;13;4;11;1.451;58
## 0.9706;13;4;11;1.451;58
## 0.9707;13;4;11;1.451;58
## 0.9708;13;4;11;1.451;58
## 0.9709;13;4;11;1.451;58
## 0.971;13;4;11;1.451;58
## 0.9711;13;4;11;1.451;58
## 0.9712;13;4;11;1.451;58
## 0.9713;13;4;11;1.451;56
## 0.9714;13;4;10;2.279;55
## 0.9715;13;4;10;2.279;55
## 0.9716;13;4;10;2.279;55
## 0.9717;14;5;8;4.861;55
## 0.9718;14;5;8;4.861;55
## 0.9719;14;5;8;4.861;55
## 0.972;14;5;8;4.861;55
## 0.9721;14;5;8;4.861;55
## 0.9722;14;5;8;4.861;55
## 0.9723;14;5;8;4.861;55
## 0.9724;14;5;8;4.861;55
## 0.9725;14;5;8;4.861;55
## 0.9726;14;5;8;4.861;55
## 0.9727;14;6;8;4.861;54
## 0.9728;14;6;8;4.861;54
## 0.9729;13;5;8;4.217;52
## 0.973;13;5;8;4.217;52
## 0.9731;13;5;8;4.217;51
## 0.9732;13;5;8;4.217;51
## 0.9733;12;4;8;3.522;49
## 0.9734;11;3;8;2.766;47
## 0.9735;11;3;8;2.766;47
## 0.9736;11;3;8;2.766;47
## 0.9737;11;3;8;2.766;47
## 0.9738;11;3;8;2.766;47
## 0.9739;11;3;8;2.766;47
## 0.974;11;3;8;2.766;47
## 0.9741;11;3;8;2.766;47
## 0.9742;11;3;8;2.766;47
## 0.9743;10;2;8;1.938;45
## 0.9744;10;2;8;1.938;45
## 0.9745;10;2;8;1.938;44
## 0.9746;10;2;8;1.938;44
## 0.9747;10;2;8;1.938;44
## 0.9748;10;2;8;1.938;44
## 0.9749;10;2;8;1.938;44
## 0.975;10;2;8;1.938;44
## 0.9751;10;2;8;1.938;44
## 0.9752;10;2;8;1.938;44
```

```
## 0.9753;10;2;8;1.938;44
## 0.9754;10;2;8;1.938;44
## 0.9755;10;2;8;1.938;44
## 0.9756;10;2;7;3.098;43
## 0.9757;10;2;7;3.098;43
## 0.9758;10;2;7;3.098;43
## 0.9759;10;2;7;3.098;43
## 0.976;10;2;7;3.098;43
## 0.9761;10;2;7;3.098;43
## 0.9762;10;2;7;3.098;43
## 0.9763;10;2;7;3.098;43
## 0.9764;10;2;7;3.098;43
## 0.9765;10;2;7;3.098;43
## 0.9766;10;2;7;3.098;43
## 0.9767;10;2;7;3.098;43
## 0.9768;10;2;7;3.098;43
## 0.9769;10;2;7;3.098;43
## 0.977;10;2;7;3.098;43
## 0.9771;11;3;7;3.926;43
## 0.9772;11;3;7;3.926;43
## 0.9773;11;3;7;3.926;43
## 0.9774;11;3;7;3.926;43
## 0.9775;11;3;7;3.926;43
## 0.9776;11;3;7;3.926;43
## 0.9777;11;4;7;3.926;42
## 0.9778;11;4;7;3.926;42
## 0.9779;11;4;7;3.926;42
## 0.978;11;4;7;3.926;42
## 0.9781;11;4;7;3.926;42
## 0.9782;11;4;7;3.926;42
## 0.9783;11;4;7;3.926;42
## 0.9784;11;4;7;3.926;42
## 0.9785;11;4;7;3.926;41
## 0.9786;11;4;7;3.926;41
## 0.9787;11;4;7;3.926;41
## 0.9788;11;4;7;3.926;41
## 0.9789;11;4;7;3.926;41
## 0.979;11;4;7;3.926;41
## 0.9791;11;4;7;3.926;41
## 0.9792;11;4;7;3.926;41
## 0.9793;11;4;7;3.926;41
## 0.9794;11;4;7;3.926;41
## 0.9795;10;3;7;3.098;39
## 0.9796;10;3;7;3.098;39
## 0.9797;10;3;7;3.098;39
## 0.9798;10;3;7;3.098;39
## 0.9799;10;3;7;3.098;39
## 0.98;10;3;7;3.098;39
## 0.9801;10;3;7;3.098;39
## 0.9802;10;3;7;3.098;39
## 0.9803;10;3;7;3.098;39
## 0.9804;11;4;6;5.265;39
## 0.9805;11;4;6;5.265;39
## 0.9806;10;3;6;4.437;37
## 0.9807;10;3;6;4.437;37
## 0.9808;10;3;6;4.437;37
## 0.9809;10;3;6;4.437;37
## 0.981;10;4;6;4.437;36
## 0.9811;10;4;6;4.437;36
## 0.9812;9;3;6;3.522;34
## 0.9813;9;3;6;3.522;34
## 0.9814;9;3;6;3.522;34
## 0.9815;9;3;6;3.522;34
## 0.9816;9;3;6;3.522;34
## 0.9817;9;3;6;3.522;34
## 0.9818;9;3;6;3.522;34
## 0.9819;9;3;6;3.522;34
## 0.982;9;3;6;3.522;34
## 0.9821;9;3;5;5.105;33
## 0.9822;9;3;5;5.105;33
## 0.9823;9;3;5;5.105;33
## 0.9824;9;3;5;5.105;33
## 0.9825;9;3;5;5.105;33
## 0.9826;9;3;5;5.105;33
## 0.9827;9;3;5;5.105;32
## 0.9828;9;3;5;5.105;32
## 0.9829;9;3;5;5.105;31
## 0.983;9;3;5;5.105;30
## 0.9831;9;3;5;5.105;30
## 0.9832;9;3;5;5.105;29
## 0.9833;9;3;5;5.105;29
## 0.9834;9;3;4;7.044;28
## 0.9835;9;3;4;7.044;28
## 0.9836;9;3;4;7.044;28
## 0.9837;9;4;4;7.044;27
## 0.9838;9;4;4;7.044;27
## 0.9839;9;4;4;7.044;27
## 0.984;9;4;4;7.044;27
## 0.9841;9;4;4;7.044;27
## 0.9842;9;4;4;7.044;27
## 0.9843;9;4;4;7.044;27
## 0.9844;9;4;4;7.044;27
## 0.9845;9;4;4;7.044;27
## 0.9846;9;4;4;7.044;27
## 0.9847;9;4;4;7.044;27
```

```
## 0.9848;9;4;4;7.044;27
## 0.9849;9;4;4;7.044;27
## 0.985;9;4;4;7.044;27
## 0.9851;9;4;4;7.044;27
## 0.9852;9;4;4;7.044;27
## 0.9853;9;4;4;7.044;27
## 0.9854;9;4;4;7.044;27
## 0.9855;9;4;4;7.044;27
## 0.9856;9;4;4;7.044;27
## 0.9857;9;4;4;7.044;27
## 0.9858;9;4;4;7.044;27
## 0.9859;9;4;4;7.044;26
## 0.986;9;4;4;7.044;26
## 0.9861;9;5;4;7.044;25
## 0.9862;9;5;4;7.044;25
## 0.9863;9;5;4;7.044;25
## 0.9864;9;5;4;7.044;25
## 0.9865;9;5;4;7.044;25
## 0.9866;9;5;4;7.044;24
## 0.9867;9;5;4;7.044;24
## 0.9868;9;5;4;7.044;23
## 0.9869;9;5;4;7.044;23
## 0.987;9;5;4;7.044;23
## 0.9871;9;5;4;7.044;23
## 0.9872;9;5;4;7.044;23
## 0.9873;9;5;4;7.044;23
## 0.9874;9;5;4;7.044;23
## 0.9875;9;5;4;7.044;23
## 0.9876;9;5;4;7.044;23
## 0.9877;9;5;4;7.044;23
## 0.9878;9;5;4;7.044;23
## 0.9879;8;4;4;6.021;21
## 0.988;8;4;4;6.021;21
## 0.9881;8;4;4;6.021;21
## 0.9882;8;4;4;6.021;21
## 0.9883;8;4;4;6.021;21
## 0.9884;8;4;4;6.021;21
## 0.9885;8;4;4;6.021;21
## 0.9886;8;4;4;6.021;21
## 0.9887;8;4;4;6.021;21
## 0.9888;8;4;4;6.021;21
## 0.9889;8;4;4;6.021;21
## 0.989;8;4;4;6.021;21
## 0.9891;8;4;4;6.021;21
## 0.9892;8;4;4;6.021;21
## 0.9893;8;4;4;6.021;21
## 0.9894;8;4;4;6.021;21
## 0.9895;8;4;4;6.021;21
## 0.9896;8;4;4;6.021;21
## 0.9897;8;5;4;6.021;20
## 0.9898;8;5;4;6.021;20
## 0.9899;8;5;4;6.021;20
## 0.99;8;5;4;6.021;20
## 0.9901;8;5;4;6.021;20
## 0.9902;8;5;4;6.021;20
## 0.9903;8;6;4;6.021;19
## 0.9904;8;6;4;6.021;19
## 0.9905;8;6;4;6.021;19
## 0.9906;8;6;4;6.021;19
## 0.9907;8;6;4;6.021;19
## 0.9908;8;6;4;6.021;19
## 0.9909;8;7;4;6.021;18
## 0.991;8;7;4;6.021;18
## 0.9911;8;7;4;6.021;18
## 0.9912;8;7;4;6.021;18
## 0.9913;8;7;4;6.021;18
## 0.9914;8;7;4;6.021;18
## 0.9915;8;7;4;6.021;18
## 0.9916;8;7;4;6.021;18
## 0.9917;8;7;4;6.021;18
## 0.9918;8;7;4;6.021;18
## 0.9919;8;7;4;6.021;18
## 0.992;8;7;4;6.021;18
## 0.9921;8;7;4;6.021;18
## 0.9922;8;7;4;6.021;18
## 0.9923;8;7;3;8.519;17
## 0.9924;8;7;3;8.519;17
## 0.9925;8;7;3;8.519;17
## 0.9926;8;7;3;8.519;17
## 0.9927;8;7;3;8.519;17
## 0.9928;8;7;3;8.519;17
## 0.9929;8;7;3;8.519;17
## 0.993;8;7;3;8.519;17
## 0.9931;8;7;3;8.519;17
## 0.9932;8;7;3;8.519;17
## 0.9933;8;7;3;8.519;17
## 0.9934;8;7;3;8.519;17
## 0.9935;8;7;3;8.519;17
## 0.9936;6;5;3;6.021;13
## 0.9937;6;5;3;6.021;13
## 0.9938;6;5;3;6.021;13
## 0.9939;5;5;2;7.959;10
## 0.994;5;5;2;7.959;10
## 0.9941;5;5;2;7.959;10
## 0.9942;5;5;2;7.959;10
```

```
## 0.9943;5;5;2;7.959;10
## 0.9944;5;5;2;7.959;10
## 0.9945;5;5;2;7.959;10
## 0.9946;4;4;2;6.021;8
## 0.9947;4;4;2;6.021;8
## 0.9948;4;4;2;6.021;8
## 0.9949;4;4;2;6.021;8
## 0.995;4;4;2;6.021;8
## 0.9951;4;4;2;6.021;8
## 0.9952;4;4;2;6.021;8
## 0.9953;4;4;2;6.021;8
## 0.9954;4;4;2;6.021;8
## 0.9955;4;4;2;6.021;8
## 0.9956;4;4;2;6.021;8
## 0.9957;3;3;2;3.522;6
## 0.9958;3;3;2;3.522;6
## 0.9959;3;3;2;3.522;6
## 0.996;3;3;2;3.522;6
## 0.9961;3;3;2;3.522;6
## 0.9962;3;3;2;3.522;6
## 0.9963;3;3;2;3.522;6
## 0.9964;3;3;2;3.522;6
## 0.9965;3;3;2;3.522;6
## 0.9966;3;3;2;3.522;6
## 0.9967;3;3;2;3.522;6
## 0.9968;3;3;2;3.522;6
## 0.9969;3;3;2;3.522;6
## 0.997;3;3;2;3.522;6
## 0.9971;3;3;2;3.522;6
## 0.9972;3;3;2;3.522;6
## 0.9973;3;3;2;3.522;6
## 0.9974;3;3;2;3.522;6
## 0.9975;3;3;2;3.522;6
## 0.9976;3;3;2;3.522;6
## 0.9977;3;3;2;3.522;6
## 0.9978;3;3;2;3.522;6
## 0.9979;3;3;2;3.522;6
## 0.998;3;3;2;3.522;6
## 0.9981;3;3;2;3.522;6
## 0.9982;3;3;2;3.522;6
## 0.9983;3;3;2;3.522;6
## 0.9984;3;3;2;3.522;6
## 0.9985;3;3;2;3.522;6
## 0.9986;3;3;2;3.522;6
## 0.9987;2;2;2;0;4
## 0.9988;2;2;2;0;4
## 0.9989;2;2;2;0;4
## 0.999;2;2;2;0;4
## 0.9991;2;2;2;0;4
## 0.9992;2;2;2;0;4
## 0.9993;2;2;2;0;4
## 0.9994;2;2;2;0;4
## 0.9995;2;2;2;0;4
## 0.9996;2;2;2;0;4
## 0.9997;2;2;2;0;4
## 0.9998;2;2;2;0;4
## 0.9999;2;2;2;0;4
## # -----------------
## # Optimum Corr Min=0.9, Max=0.9923, Mean=0.9
## Getting the clusters of the peaks...
## Matching clusters with reference metabolites...
## Done.
```

3. Example of results that can be obtained for each cluster formed:

- Peaks of the cluster:

Hide

```
nmr_ID_res$Cluster1$cluster.peaks[,1]
```

```
##  [1] 0.10 0.30 0.59 0.75 1.20 1.35 1.51 2.31 2.34 2.40 2.64 2.67 2.70 2.88 3.00 3.03 3.06 3.09 3.12 3.15 3.18 3.21 3.24 3.27 3.29 3
.33 3.36 3.39 3.42
## [30] 3.45 3.60 4.02 4.05 4.08 4.13 4.17 4.20 4.25 4.28 4.31 4.34 4.38 4.41 4.45 4.50 4.53 4.55 4.58 4.63 4.66 4.71 4.74 5.45 6.20 6
.56 6.62 6.70 6.75
## [59] 6.81 6.88 6.92 6.96 7.03 7.09 7.56 7.71 7.74 7.81 7.84 7.87 7.91 7.96 8.14 9.29 9.51 9.66
```

- Summary of the top metabolites identified for that cluster, with the respective scores:

Hide

```
knitr::kable(nmr_ID_res$Cluster1$summary, col.names="Jaccard Index")
```

	Jaccard Index
HMDB0001197	0.1798
HMDB0001248	0.1798
HMDB0000033	0.1758
HMDB0001849	0.1650
HMDB0001885	0.1625
HMDB0001434	0.1605
HMDB0001932	0.1489

	Jaccard Index
HMDB0000472	0.1446
HMDB0000866	0.1446
HMDB0000217	0.1429

4. Further results for each metabolite matched:

- Score (jaccard index)

Hide

```
nmr_ID_res$Cluster1$metabolites.matched[[1]]$score
```

```
## [1] 0.1798
```

- Peaks of the reference spectrum that matched the cluster

Hide

```
nmr_ID_res$Cluster1$metabolites.matched[[1]]$matched_peaks_ref
```

```
##  [1] 2.32 2.39 4.02 4.04 4.05 4.28 4.29 4.31 4.34 4.37 4.40 4.45 4.47 4.50 7.54 7.85
```

- Peaks of the cluster that matched the reference spectrum

Hide

```
nmr_ID_res$Cluster1$metabolites.matched[[1]]$matched_peaks_clust
```

```
##  [1] 2.31 2.40 4.02 4.05 4.08 4.25 4.28 4.31 4.34 4.38 4.41 4.45 4.50 4.53 7.56 7.84
```

- Peaks of the reference spectra

Hide

```
nmr_ID_res$Cluster1$metabolites.matched[[1]]$reference_peaks
```

```
##  [1] 2.32 2.39 3.88 3.89 3.90 3.90 4.02 4.04 4.05 4.06 4.06 4.08 4.09 4.28 4.29 4.31 4.34 4.37 4.40 4.45 4.46 4.47 4.49 4.50 4.51 5
.82 5.83 7.54 7.60
## [30] 7.85 8.30
```

16 Pathway Analysis

16.1 Functions to use

1. Get only the paths of the mentioned organism that contain one or more of the given compounds

> get_paths_with_cpds_org(organism_code, compounds, full.result=T)

- *organism_code*: Organism code. The correct code for an organism can be consulted using function *get_OrganismsCodes*, mentioned in the additional functions below;
- *compounds*: Named vector with kegg codes of compounds and respective names. This vector can be obtained by using the function *get_cpd_names* or the function *convert_hmdb_to_kegg*, mentioned in the additional functions below;
- *full.result*: boolean value (TRUE or FALSE) indicating if a full result is to be given. Defaults to TRUE.

If full result is chosen, the data frame returned contains information on the pathways of the organism that contains one or more of the given compounds and, for each pathway, the kegg codes (and their names) of the compounds given that are present in that path.

If full result is not wanted, only the pathways will be given.

2. Create the metabolic pathway wanted.

The pathway created contains the compounds, reactions and other paths that it connects to as nodes.

The compounds given in compounds are colored in blue, while the rest of the compounds are colored in grey.

The other paths that it may connect to are colored in orange.

Reversive reactions are colored in green and the irreversible ones in red.

> pathway_analysis(compounds, pathway, nodeNames="kegg", nodeTooltip=F, map.zoom=F, map.layout="preset", map.width=NULL, map.height=NULL)

- *compounds*: vector of compounds of interest;
- *pathway*: KEGG code (e.g., "hsa00010") of the path wanted;
- *nodeNames*: how the nodes should be named. If "kegg", nodes are named with kegg codes. If "names", nodes are named with the common names.
- *nodeTooltip*: if a tooltip should appear when hovering a node. Only works in certain environments;
- *map.zoom*: if the map should have the zoom in and out option. Only works in certain environments;
- *map.layout*: layout of the map, available values are the ones of cytoscape ("breadthfirst", "preset", "cose", ...);
- *map.width*: width of the map, in percentage;
- *map.height*: Height of the map, in px (e.g. "500px").

16.2 Additional functions

The following function returns a dataframe with the t-number, organism kegg code, full name and phylogeny for each kegg organism:

> get_OrganismsCodes()

The following function returns a named vector with the names of the compounds. The names of the vector are the compounds' names and the vector elements the

kegg codes.

> get_cpds_names(kegg_codes)

- *kegg_codes*: Character vector with kegg codes.

The following function return a named vector with kegg codes and respective metabolite names. Vector names are the compound names and the vector elements the kegg codes:

> convert_hmdb_to_kegg(hmdb_codes)

- *hmdb_codes*:Vector with the HMDB codes (each hmdb code must have 7 digits, e.g., HMDB0000001).

The following function returns the metabolic kegg paths present in mentioned organism:

> get_MetabPaths_org(org_code)

- *org_code*: Organism code. The correct code for an organism can be consulted using function *get_OrganismsCodes*, mentioned above.

16.3 Example

This example uses the results obtained in the example for identification of metabolites from NMR peaks (here).

1. Get identified metabolites in one vector:

Hide

```
id_metabs=c()
for (cluster in nmr_ID_res){
  id_metabs=c(id_metabs, names(cluster$summary))
}
id_metabs
```

```
##  [1] "HMDB0001197" "HMDB0001248" "HMDB0000033" "HMDB0001849" "HMDB0001885" "HMDB0001434" "HMDB0001932" "HMDB0000472" "HMDB0000866"
"HMDB0000217"
## [11] "HMDB0000921" "HMDB0000077" "HMDB0000032" "HMDB0000501" "HMDB0001830" "HMDB0000937" "HMDB0000761" "HMDB0000871" "HMDB0000374"
"HMDB0000899"
## [21] "HMDB0000217" "HMDB0000235" "HMDB0001878" "HMDB0001569" "HMDB0000920" "HMDB0000243" "HMDB0000239" "HMDB0000225" "HMDB0000355"
"HMDB0000426"
## [31] "HMDB0000720" "HMDB0000474" "HMDB0000635" "HMDB0000661" "HMDB0001545" "HMDB0003282" "HMDB0002024" "HMDB0000076" "HMDB0000434"
"HMDB0000440"
## [41] "HMDB0001930" "HMDB0000462" "HMDB0001874" "HMDB0001238" "HMDB0001389" "HMDB0000115" "HMDB0000639" "HMDB0001511" "HMDB0000863"
"HMDB0001847"
## [51] "HMDB0000574" "HMDB0000168" "HMDB0000187" "HMDB0000707" "HMDB0001316" "HMDB0000017" "HMDB0000034" "HMDB0000300" "HMDB0000444"
"HMDB0001847"
## [61] "HMDB0000484" "HMDB0001046" "HMDB0000738" "HMDB0000197" "HMDB0000840" "HMDB0001860" "HMDB0000290"
```

2. Convert metabolites from HMDB codes to KEGG codes:

Hide

```
id_metabs_kegg=convert_hmdb_to_kegg(id_metabs)
```

```
## The following hmdb code is not available at kegg:  HMDB0001885
## The following hmdb code is not available at kegg:  HMDB0001434
## The following hmdb code is not available at kegg:  HMDB0000501
## The following hmdb code is not available at kegg:  HMDB0000720
## The following hmdb code is not available at kegg:  HMDB0000635
## The following hmdb code is not available at kegg:  HMDB0000434
## The following hmdb code is not available at kegg:  HMDB0000444
## The following hmdb code is not available at kegg:  HMDB0001046
```

Hide

```
#id_metabs_kegg
knitr::kable(cbind(names(id_metabs_kegg), id_metabs_kegg), row.names=F, col.names=c("Names", "KEGG codes"))
```

Names	KEGG codes
FADH	cpd:C01352
FAD	cpd:C00016
Carnosine	cpd:C00386
Propranolol	cpd:C07407
Metoprolol	cpd:C07202
5-Hydroxy-L-tryptophan	cpd:C01017
N-Acetyl-L-tyrosine	cpd:C01657
NADP	cpd:C00006
Cholestenone	cpd:C00599
Dehydroepiandrosterone	cpd:C01227
7-Dehydrocholesterol	cpd:C01164
Progesterone	cpd:C00410
Stigmasterol	cpd:C05442
Lithocholic acid	cpd:C03990
5alpha-Cholestanone	cpd:C03238

Names	KEGG codes
17-Hydroxyprogesterone	cpd:C01176
Androstanedione	cpd:C00674
NADP	cpd:C00006
Thiamine	cpd:C00378
Thymol	cpd:C09908
Epi-coprostanol	cpd:C12978
11a-Hydroxyprogesterone	cpd:C03747
Pyruvic acid	cpd:C00022
Pyridoxine	cpd:C00314
Oxoadipic acid	cpd:C00322
3-Hydroxymethylglutaric acid	cpd:C03761
Citramalic acid	cpd:C00815
Butanone	cpd:C02845
Glutaric acid	cpd:C00489
Pyridoxal	cpd:C00250
1-Methylguanine	cpd:C04152
Imidazoleacetic acid	cpd:C02835
Dihydrouracil	cpd:C00429
3-Hydroxyphenylacetic acid	cpd:C05593
Ranitidine	cpd:D00673
Allantoin	cpd:C01551
D-threo-Isocitric acid	cpd:C00451
N-Acetylserotonin	cpd:C00978
Melatonin	cpd:C01598
Glycolic acid	cpd:C00160
Galactaric acid	cpd:C00879
Phosphocreatine	cpd:C02305
Isopropyl alcohol	cpd:C01845
Caffeine	cpd:C07481
L-Cysteine	cpd:C00097
L-Asparagine	cpd:C00152
L-Serine	cpd:C00065
4-Hydroxyphenylpyruvic acid	cpd:C01179
6-Phosphogluconic acid	cpd:C00345
4-Pyridoxic acid	cpd:C00847
Adenine	cpd:C00147
Uracil	cpd:C00106
Caffeine	cpd:C07481
Vanillic acid	cpd:C06672
Indole	cpd:C00463
Indoleacetic acid	cpd:C00954
Salicyluric acid	cpd:C07588
Paraxanthine	cpd:C13747
Uridine diphosphate-N-acetylglucosamine	cpd:C00043

3. Get the metabolic paths that have one ore more of the identified compounds, obtaining the full results

Hide

```
cpds_paths=specmine::get_paths_with_cpds_org("ath", id_metabs_kegg)
```

```
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
## No encoding supplied: defaulting to UTF-8.
```

Hide

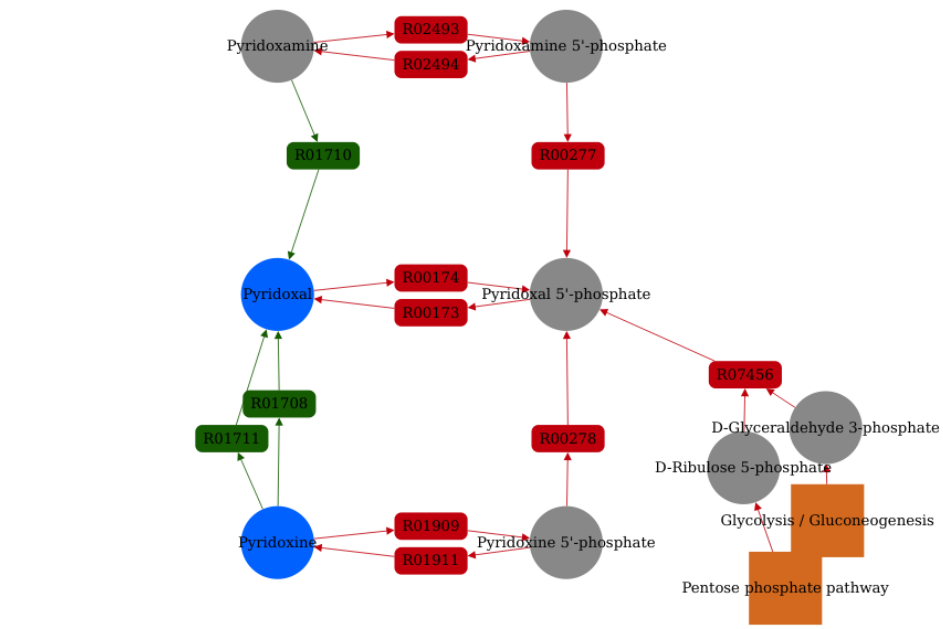
```
knitr::kable(cpds_paths)
```

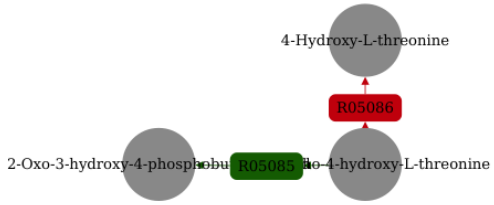
	pathways	compounds	compounds_names
Glycolysis / Gluconeogenesis	ath00010	cpd:C00022	Pyruvic acid
Citrate cycle (TCA cycle)	ath00020	cpd:C00022	Pyruvic acid
Pentose phosphate pathway	ath00030	cpd:C00345	6-Phosphogluconic acid
Steroid biosynthesis	ath00100	cpd:C01164; cpd:C05442	7-Dehydrocholesterol; Stigmasterol
Ubiquinone and other terpenoid-quinone biosynthesis	ath00130	cpd:C01179	4-Hydroxyphenylpyruvic acid
Purine metabolism	ath00230	cpd:C00147	Adenine
Caffeine metabolism	ath00232	cpd:C13747	Paraxanthine
Pyrimidine metabolism	ath00240	cpd:C00429; cpd:C00106	Dihydrouracil; Uracil
Alanine, aspartate and glutamate metabolism	ath00250	cpd:C00152; cpd:C00022	L-Asparagine; Pyruvic acid
Glycine, serine and threonine metabolism	ath00260	cpd:C00065; cpd:C00022	L-Serine; Pyruvic acid
Monobactam biosynthesis	ath00261	cpd:C00022	Pyruvic acid
Cysteine and methionine metabolism	ath00270	cpd:C00065; cpd:C00097; cpd:C00022	L-Serine; L-Cysteine; Pyruvic acid
Valine, leucine and isoleucine biosynthesis	ath00290	cpd:C00022	Pyruvic acid
Lysine degradation	ath00310	cpd:C00322	Oxoadipic acid
Histidine metabolism	ath00340	cpd:C02835	Imidazoleacetic acid
Tyrosine metabolism	ath00350	cpd:C01179; cpd:C00022	4-Hydroxyphenylpyruvic acid; Pyruvic acid
Tryptophan metabolism	ath00380	cpd:C00978; cpd:C00322; cpd:C01598; cpd:C00954	N-Acetylserotonin; Oxoadipic acid; Melatonin; Indoleacetic acid
Phenylalanine, tyrosine and tryptophan biosynthesis	ath00400	cpd:C00463; cpd:C01179	Indole; 4-Hydroxyphenylpyruvic acid
beta-Alanine metabolism	ath00410	cpd:C00429; cpd:C00106	Dihydrouracil; Uracil
Cyanoamino acid metabolism	ath00460	cpd:C00097; cpd:C00152; cpd:C00065	L-Cysteine; L-Asparagine; L-Serine
Glutathione metabolism	ath00480	cpd:C00006; cpd:C00097	NADP; L-Cysteine
Amino sugar and nucleotide sugar metabolism	ath00520	cpd:C00043	Uridine diphosphate-N-acetylglucosamine
Sphingolipid metabolism	ath00600	cpd:C00065	L-Serine
Pyruvate metabolism	ath00620	cpd:C00022	Pyruvic acid
Glyoxylate and dicarboxylate metabolism	ath00630	cpd:C00160; cpd:C00065	Glycolic acid; L-Serine
Butanoate metabolism	ath00650	cpd:C00022	Pyruvic acid
C5-Branched dibasic acid metabolism	ath00660	cpd:C00022	Pyruvic acid
Carbon fixation in photosynthetic organisms	ath00710	cpd:C00022	Pyruvic acid
Thiamine metabolism	ath00730	cpd:C00097; cpd:C00378; cpd:C00022	L-Cysteine; Thiamine; Pyruvic acid
Riboflavin metabolism	ath00740	cpd:C00016	FAD
Vitamin B6 metabolism	ath00750	cpd:C00314; cpd:C00250	Pyridoxine; Pyridoxal
Nicotinate and nicotinamide metabolism	ath00760	cpd:C00006	NADP
Pantothenate and CoA biosynthesis	ath00770	cpd:C00429; cpd:C00022; cpd:C00097; cpd:C00106	Dihydrouracil; Pyruvic acid; L-Cysteine; Uracil
Terpenoid backbone biosynthesis	ath00900	cpd:C00022	Pyruvic acid
Zeatin biosynthesis	ath00908	cpd:C00147	Adenine
Sulfur metabolism	ath00920	cpd:C00065; cpd:C00097	L-Serine; L-Cysteine
Isoquinoline alkaloid biosynthesis	ath00950	cpd:C01179	4-Hydroxyphenylpyruvic acid
Aminoacyl-tRNA biosynthesis	ath00970	cpd:C00152; cpd:C00097; cpd:C00065	L-Asparagine; L-Cysteine; L-Serine

4. Visualization of one of the pathways obtained in the previous point (Vitamin B6)

Hide

```
pathway_analysis(id_metabs_kegg, "ath00750", nodeNames="names", map.zoom=T, nodeTooltip=T)
```





17 Other Functions

You can convert the absorbance values of the dataset into transmittance values, and vice versa

```
absorbance_to_transmittance(dataset)
```

```
transmittance_to_absorbance(dataset, percent = T)
```

You can count the number of missing values on the whole dataset, and per sample or variable

```
count_missing_values(dataset)
```

```
count_missing_values_per_sample(dataset)
```

```
count_missing_values_per_variable(dataset)
```

You can get the data matrix on the dataset

```
get_data(dataset)
```

You can get the data on the dataset as a data frame

```
get_data_as_df(dataset)
```

You can get the values of a variable in a sample

```
get_data_value(dataset, x.axis.val, sample, by.index = F)
```

- *dataset*: a specmine dataset;
- *x.axis.val*: string indicating the name of the variable or numeric value indicating the position (index) of the variable in the data;
- *sample*: string indicating the name of the sample or numeric value indicating the position (index) of the sample in the data;
- *by.index*: boolean value (TRUE or FALSE) indicating if in the arguments *x.axis.val* and *sample* is given the index (by.index=TRUE) or the name of the variable and sample (by.index=FALSE). Defaults to FALSE.

You can all samples' values in the dataset for a certain variable

```
get_data_values(dataset, x.axis.val, by.index = FALSE)
```

- *dataset*: a specmine dataset;
- *x.axis.val*: string indicating the name of the variable or numeric value indicating the position (index) of the variable in the data;
- *by.index*: boolean value (TRUE or FALSE) indicating if in the argument *x.axis.val* is given the index (by.index=TRUE) or the name of the variable (by.index=FALSE). Defaults to FALSE.

You can get a data frame of the metadata of the dataset

```
get_metadata(dataset)
```

You can get the value of a metadata variable in a sample

```
get_metadata_value(dataset, variable, sample)
```

- *dataset*: a specmine dataset;
- *variable*: string indicating the name of the metadata variable or numeric value indicating the position (index) of the metadata variable;
- *sample*: string indicating the name of the sample or numeric value indicating the position (index) of the sample;

You can get the samples' values of a metadata variable

```
get_metadata_var(dataset, var)
```

- *dataset*: a specmine dataset;
- *variable*: string indicating the name of the metadata variable or numeric value indicating the position (index) of the metadata variable.

You can get the names of the samples on the dataset

```
get_sample_names(dataset)
```

You can get the type of dataset

```
get_type(dataset)
```

You can get the data x values as numbers instead of strings. Only possible if not concentrations data

```
get_x_values_as_num(dataset)
```

You can get the data x values as strings

```
get_x_values_as_text(dataset)
```

You get know if the dataset in question is spectral or not

```
is_spectra(dataset)
```

You can update the metadata in the dataset

```
set_metadata(dataset, new.metadata)
```

- *dataset*: a specmine dataset;
- *new.metadata*: matrix or dataframe of the new metadata.

You can give new names to the samples in the dataset

```
set_sample_names(dataset, new.sample.names)
```

- *dataset*: a specmine dataset;
- *new.samples.names*: character vector with the new names for the samples.

You can set a new label to the yy axis

```
set_value_label(dataset, new.val.label)
```

You can set a new label to the xx axis

```
set_x_label(dataset, new.x.label)
```

You can give new values to the xx axis

```
set_x_values(dataset, new.x.values, new.x.label = NULL)
```

- *dataset*: a specmine dataset;
- *new.x.values*: numeric or character vector with the new values for the xx axis;
- *new.x.label*: string indicating the new label for the xx axis. **optional**

18 Problems running code: solutions

18.1 "Maximum number of DLLs reached …"

If you receive this error message, you have to increase the maximum number of DLLs R can load by setting the variable R_MAX_NUM_DLLS.

1. Find out where the Renviron or Renviron.site files are located in your machine:

Hide

```
dir(Sys.getenv("R_HOME"),recursive = T,full.names = T,pattern = "Renviron")
```

2. Add the command *R_MAX_NUM_DLLS=150* at the end of one of these files:

- Example for MACOS:

Hide

```
system(' echo "R_MAX_NUM_DLLS=150" >> /Library/Frameworks/R.framework/Resources/etc/Renviron')
```

- Example for UBUNTU:

Command has to be run as administrator. So run r on terminal (type sudo R), insert password and type:

Hide

```
system(' echo "R_MAX_NUM_DLLS=150" >> /usr/lib/R/etc/Renviron')
```

18.2 If you are getting a problem with *rgl* package

If you are on linux and having problems using certain functions due to *rgl* package ("*object 'rgl_clear' not found*", for example), the problem may be the version of the *rgl* package you have.

So, performing the following in the terminal may be necessary to use these functions:

```
sudo apt-get install libglu1-mesa-dev
```

After this, reinstall the *rgl* package

Hide

```
install.packages("rgl")
```

18.3 *Error in mvrValstats(object = object, estimate = "train") : could not find function "mvrValstats"*

When runing the function *train_models_performance*, the above error may arise.

Two possible solutions are:

- Reinstall *caret* package, as it might be because you have an older version of this package, where the bug was fixed already;

Hide

```
install.packages("caret")
```

- Or, in case the error persists, you have to load the *pls* package, as a new release of *caret* package without the bug was not yet provided.

Hide

```
library(pls)
```