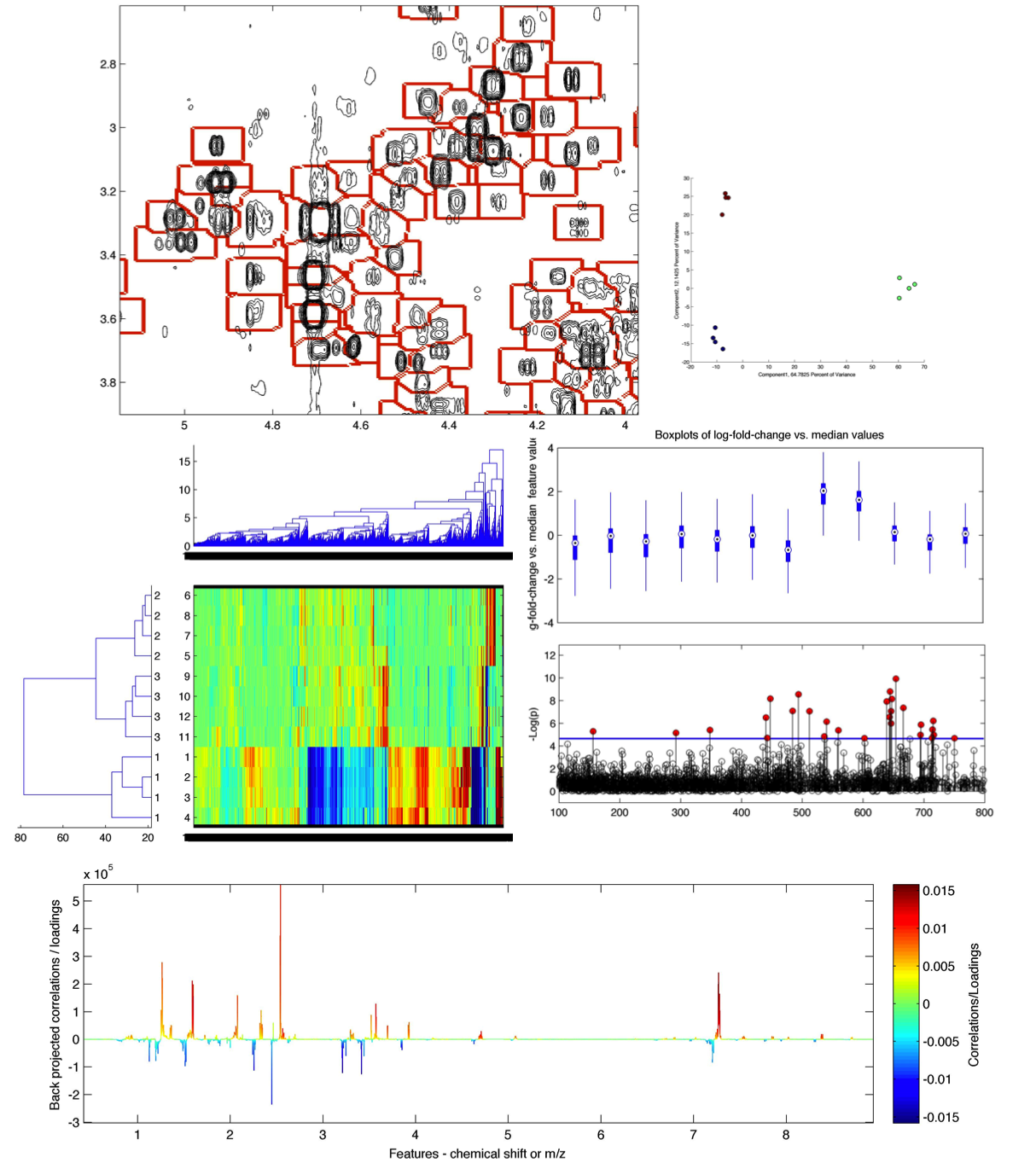
Metabolomics Toolbox for Matlab



Steven Robinette

PhD Candidate, Imperial College London

NMR and MS dataset contributed by

Yevgeniy Izrayelit, PhD Candidate, Schroeder Lab, Cornell University

**Preprocess 1D NMR**

spectra=Load1D(path,format)

[X,ppm,XTitles]=Setup1D(spectra,shiftpoints,resolution)

XR=remove\_region(X,ppm,shift1,shift2)

XAL=star\_align1D(X,ppm,represent,alignment\_method,Seg\_ppm,MaxShift\_ppm,slack\_ppm)

XALg=guide\_align1D(X,ppm,cluster\_method,alignment\_method)

[peakmatrix,shifts]=Peakpick1D(X,ppm,represent,peakthresh,mode)

displaypeak1D(X,ppm,shift,Y)

**Preprocess 2D NMR**

spectra=Load2D(path)

vis2D(spectra,disp,thresh)

Setup2D(spectra,shiftpoints1,shiftpoints2)

stackplot(X,XNoise,ppm1,ppm2,Y,thresh)

label=segment2D(spectset,noiseset,ppm1,ppm2,STNthresh,ppmbetweenNuc1,ppmbetweenNuc2)

XAL=star\_align2D(X,label,ppm1,ppm2,maxshift)

XALg=HATS(X,XNoise,ppm1,ppm2,label,maxshift)

binmat=bin2D(X,label)

displaypeak2D(X,XNoise,label,indices,ppm1,ppm2)

**Load XCMS**

importXCMS(filename)

**Data Normalization/Stabilization**

normcheck(X)

[XN,factors]=normalize(X,method,features)

varcheck(X,features)

XS=scale(X,method,offset)

**Data Analysis/Pattern Recognition**

PCA=nipalsPCA(X,components)

PLS=plsPV(Y,matrix,nfold,type,permutations)

[corrcoefs,p]=statsy(X,target,method)

[p,sigs]=MWAS(X,Y,correction)

[sample\_order,variable\_order]=two\_way\_cluster(X,algorithm,cluster\_metric,Y)

VisScores(X,model,components,Y)

VisLoadings1D(X,loadings,features,range)

[corr,covarX]=VisLoadings2D(X,loadings,ppm1,ppm2,disp,label)

manhattan(X,Y,features,p,sigs,correction)

**Function Reference:**

spectra=Load1D(path,format)

Loads all Bruker NMR data from the processed 1r file at 'path'.

Arguments:

path Path to the directory with spectral directories

format Data format- 'bruker' is supported

Return Values:

in the form of spectra.\*

real Real part of the spectra.

imaginary Imaginary part of the spectra (may not be avaliable).

ppm The PPM scale.

XTitles the titles of the spectral directories

[X,ppm,XTitles]=Setup1D(spectra,shiftpoints,resolution)

Preprocess Bruker NMR data from array spectra. This function

assembles data matrix X from array spectra.

Arguments:

spectra Array of spectra produced by Load1D.m

shiftpoints (Optional)- ppm domain range ie [10.5 -0.5]

resolution Data resolution, ie 65536 for 64k data

Return Values:

X Spectral matrix, aligned and normalized if specified

ppm Chemical shift vector

Titles Titles of NMR spectra

XR=remove\_region(X,ppm,shift1,shift2)

Removes region between ppm value of "shift1" and ppm value of "shift2" such as water region.

Arguments:

X Data matrix of spectra

ppm Chemical shift vector corresponding to X

shift1 upfield ppm of region to be removed

shift2 downfield ppm of region to be removed

Return Values:

XR Spectral matrix, with region removed

XAL=star\_align1D(X,ppm,represent,alignment\_method,Seg\_ppm,MaxShift\_ppm,slack\_ppm)

Calculates star alignment to a representative spectrum using

alignment method of your choice - default is PAFFT

Arguments:

X Data matrix of spectra

ppm chemical shift vector

represent representative spectrum of spectral set- can be 'mean','median','max', 'min', 'var', or an interger for the index of the spectrum in the full stack

alignment\_method string, either 'CCOW','ICOSHIFT','RAFFT','PAFFT'

Optional Arguments:

Seg\_ppm Length of segment in ppm (dafault 0.08)

MaxShift\_ppm Maximum distance a segment can be shifted in ppm (default is 0.05)

slack\_ppm Slack parameter for CCOW in ppm- default is 0.005

Return Values:

XAL Star-aligned spectra

XALg=guide\_align1D(X,ppm,cluster\_method,alignment\_method)

Calculates guided alignment using hierarchical clustering and an

alignment method of your choice - default is PAFFT

Arguments:

X Data matrix of spectra

ppm chemical shift vector

cluster\_method string, either 'correlation' or 'spearman'

alignment\_method string, either 'CCOW','ICOSHIFT','RAFFT','PAFFT'

Return Values:

XALg Guide-aligned spectra

[peakmatrix,shifts]=Peakpick1D(X,ppm,represent,peakthresh,mode)

Detects peaks in 1D NMR spectra and constructs matrix of peak intensities with chemical shifts of each peak maximum.

Arguments:

X Data matrix of spectra

ppm Chemical shift vector corresponding to X

represent representative spectrum of spectral set- can be

'mean','median','max', 'min', 'var', or an interger for the index of the spectrum in the full stack

peakthresh values between 0 and 1, higher threshold is greater specificity

mode 'Simple' (multiple of noise), or 'Complex'

Return Values:

peakmatrix Matrix of peak intensities

shifts vector of chemical shifts at each position in

peakmatrix

spectra = Load2D(path)

Load Bruker NMR data from the processed 2rr file at 'path'.

Arguments:

path Path to the 2rr files to be loaded.

Return Values:

in the form of spectra.\*

real Real part of the spectra.

ppm1 F2 PPM scale.

ppm2 F1 PPM scale.

vis2D(spectra,disp,thresh)

Display plot of spectrum after loading spectra.

Arguments:

spectra Index of spectra to see: ex. spectra(1)

disp 'f' for fast, 'c' for contour

thresh Contour/color threshold- try 5 to start

Setup2D(spectra,shiftpoints)

Create 3D matrix of Bruker NMR data from array of spectra. This function assembles data matrix X from array spectra.

Arguments:

spectra Array of spectra produced by Stack2D.m

shiftpoints1 Specify F2 chemical shift range eg; [-1 10] for 1H

shiftpoints2 Specify F1 chemical shift range eg; [-5 170] for 13C

Return Values:

X Spectral matrix

XTitles Bruker format spectrum titles

XNoise Local Noise matrix

ppm1 Chemical shift vector along F2

ppm2 Chemical shift vector along F1

stackplot(X,XNoise,ppm1,ppm2,Y,thresh)

Plots stacked contour plot of X colored by Y (if known).

Arguments:

X Data matrix of spectra

XNoise Calculated noise matrix

ppm1 Chemical shift vector of F2

ppm2 Chemical shift vector of F1

Y if response vector Y is known, use it to color the contour plots (optional)

thresh Intensity threshold- can be scalar or vector; if vector

same number of elements as size(X,3)

label=segment2D(X,XNoise,ppm1,ppm2,STNthresh,ppmbetweenNuc1,ppmbetweenNuc2)

Spectral segmentation algorithm using Koradi noise calculation and

morphological operations to grow segments - as outlined in HATS and

mvaDANS

Arguments:

X Data matrix of spectra

XNoise Data matrix of spectral noise from Koradi algorithm

ppm1 Chemical shift vector of F2

ppm2 Chemical shift vector of F1

STNthresh Signal to noise threshold for detection - default 10 for homonuclear, 4 for heteronuclear

ppmbetweenNuc1 allowed distance between maxima in same crosspeak in F2 dimension, default 0.035 for homonuclear, 0.025 for heteronuclear

ppmbetweenNuc2 allowed distance between maxima in same crosspeak in F1 dimension, default 0.035 for homonuclear, 0.3 for heteronuclear

Return Values:

label matrix equal in size to spectrum with index values for each region

XAL=star\_align2D(X,label,ppm1,ppm2,maxshift)

Star alignment against mean spectrum using local alignment function from HATS.

Arguments:

X Data matrix of spectra

label label matrix from segment2D.m

ppm1 Chemical shift vector of F2

ppm2 Chemical shift vector of F1

maxshift Maximum shift if F2 and F1 dimensions [F2,F1]

Return Values:

XAL Aligned spectral matrix

XALg=HATS(X,XNoise,ppm1,ppm2,label,maxshift)

Alignment using HATS method. Robinette et al. Analytical Chemistry,

2011. 83(5):1649-1657. Applies segment2D if no label matrix is provided

Arguments:

X Data matrix of spectra

XNoise Local Noise matrix

ppm1 Chemical shift vector of F2

ppm2 Chemical shift vector of F1

label label matrix from segment2D.m (optional)

maxshift Maximum shift if F2 and F1 dimensions [F2,F1] (optional)

Return Values:

XALg Aligned spectral matrix

binmat=bin2D(X,XNoise)

Automatic binning of 3D spectral matrix X using spectral segments in

label matrix

Arguments:

X Spectral matrix

label Spectral segmentation matrix

Return Values:

binmat Matrix of bin intensities

displaypeak2D(X,XNoise,label,indices,ppm1,ppm2)

Inputs:

X Data matrix of spectra

XNoise Local Noise matrix

label label matrix from segment2D.m

indices index in label matrix of peak to display - eg. [1] or

find(sigs==1)

ppm1 Chemical shift vector of F2

ppm2 Chemical shift vector of F1

displaypeak2D plots a set of bin regions on the mean pseudo-spectrum of X. Use this script to assign correlated bin regions.

importXCMS(filename)

Import .xls file of XCMS processed MS data.

Arguments:

filename Path to .xls file containing XCMS output data

Outputs:

msmatrix N x P matrix of raw MS data - abundances of each feature

P for all N spectra processed by XCMS

massvect Median m/z value for each feature P

RTvect Median retention time for each feature P

metadata XCMS calculated metadata, {fold change, t-stat, p-value, median m/z, min m/z, max m/z, median RT, min RT, max RT, number of spectra with feature, number of class 1 spectra with feature, number of class 2 spectra with feature

normcheck(X)

Displays histogram and box plots of log-fold change vs. median for all features in each spectrum. Dilution / normalization effects are often visible as distributions not centered at 0.

Arguments:

X N x P matrix of spectral features for each sample

[XN,factors]=normalize(X,method,features)

Normalize spectral matrix X to total area, single feature, or integral of set of features.

Arguments:

X N x P matrix of spectral features for each sample

method 'total' for Total Area, 'PQN' for Probablistic Quotient, 'intensity' for normalization to single feature, 'integral' for normalization to sum of set of features

features only required for 'intensity' or 'integral'. For

'intensity', the index of the feature in X to normalize to - e.g. [10] for X(:,10). For 'integral', the range of features in X that span the peak to normalize to – eg [10,50] for X(:,10:50)

Outputs:

XN N x P matrix of normalized data

factors N calculated normalization factors: XN = X / factors

varcheck(X,features)

Displays histogram and box plots of log-fold change vs. median for all features in each spectrum. Dilution / normalization effects are often visible as distributions not centered at 0.

Arguments:

X N x P matrix of spectral features for each sample

features 1 x N vector of spectral features - chemical shifts or m/z

XS=scale(X,method,offset)

Normalize spectral matrix X to total area, single feature, or integral of set of features.

Arguments:

X N x P matrix of spectral features for each sample

method 'log' for log2 fold-change vs. median scaling, 'logoff' for offset log2 fold-change vs. median, 'mc' for mean-centering (only), 'auto' for autoscaling mean-center and univariance scaling), 'pareto' for pareto scaling. 'integral' for normalization to sum of set of features

offset optional feature for 'log' or 'logoff'. 'log' uses

default offset of 0.0001 to prevent InF, 'logoff' uses automatically calculated median.

Outputs:

XS N x P matrix of scaled data

PCA=nipalsPCA(X,components)

Calculates first 'components' principal components of X by NIPALS algorithm and outputs an array with scores, loadings, and variance for each principal component

Arguments:

X N x P matrix of spectral features for each sample

components scalar, number of components to calculate

Outputs:

PCA.scores Scores for each spectrum in X along each principal component

PCA.loadings Coefficients of each variable contributing to generation of the scores in each principal component.

PCA.variance Total variance accounted for by each principal component

PLS=plsPV(X,Y,nfold,type,permutations,method)

Calculates a partial least squares regression of Y on X using PLS components of X by SIMPLS algorithm and outputs an array with scores, and loadings for each PLS component and the PLS coefficients vector. This function uses nfold permutation-validation to avoid overfitting, and the user will be asked to select the optimal number of PLS components to avoid over- and under-fitting the data. This

function can perform PLS regression or PLS discriminant analysis.

Scaling is performed during validation in order to avoid the introduction of bias.

Arguments:

X Data matrix from spectra: binmat or full X matrix

Y Response vector taking real values in the case of regression or dummy variables (ie, ones and zeros) in the case of 2 class discriminant analysis

nfold N-fold cross validation: ie, 5 specifies five-fold cross validation which splits X up into 5 sets of 80% training and 20% validation data. The model constructed from the training data is evaluated on the validation data and R2 and Q2 values are returned.

type 're' for regression, or 'da' for discriminant analysis

permutation Number of permutations to test cross-validation.

method (Optional) - 'log' for log2 fold-change vs. median scaling, 'logoff' for offset log2 fold-change vs. median, 'mc' for mean-centering (only), 'auto' for autoscaling (mean-center and univariance scaling), 'pareto' for pareto scaling. 'integral' for normalization to sum of set of features

Return Values:

PLS.scores Scores for each spectrum in X along each PLS component

PLS.loadings Coefficients of each variable contributing to generation of the scores in each PLS component.

PLS.beta PLS coefficients constructed from user selected optimal number of PLS components

PLS.r2 Fit of model to training data

PLS.q2 Fit of model to test data

PLS.variance Variance in X for each component

[corrcoefs,p]=statsy(X,target,method)

Statistical Spectroscopy - Calculates correlation coefficients between features in spectral matrix 'X' and feature 'target'. 'Target' can be either an index of a feature in X, or a vector or features of size (X,1). When applied to NMR data, this is STOCSY. When applied between NMR and MS, this is SHY.

Arguments:

target Chemical shift of driver peak or response vector with length equal to size(X,1)

X Data matrix of spectra

method Correlation metric, 'Pearson', 'Spearman', or 'Kendall'.

Default 'Pearson'

[p,sigs]=MWAS(X,Y,correction)

Metabolome-wide association study design; t-test or logistic regression of spectral features against classes Y. Correction for multiple hypothesis testing using either Sidak or Bonferroni correction.

Arguments:

X Data matrix of spectra

Y Vector of classes (0 and 1)

correction Either Bonferroni, 'bonferroni', or Sidak, 'sidak'

Ouputs:

p P values for t-test or logistic regression model

sigs logical vector of significant features

[sample\_order,variable\_order]=two\_way\_cluster(X,algorithm,cluster\_metric,Y)

Two-way hierarchical clustering of data matrix 'X' using clustering

algorithm 'algorithm' and distance metric 'cluster\_metric'. This

function outputs order vectors for observations and variables along with an image of the clustered matrix. See linkage.m for more information about 'algorithm' and 'cluster\_metric' parameters.

Arguments:

X Data matrix of spectra

algorithm Either 'average', 'centroid', 'complete', 'median',

'single', 'ward', or 'weighted'.

cluster\_metric See pdist.m for more options. Use 'euclidean',

'correlation', or 'spearman' as defaults.

Y Optional: Vector of classes

VisScores(X,model,components,Y)

Displays PCA or PLS model scores plots for 1-3 components specified in vector "components"

Arguments:

X Data matrix of spectra

model Model structure generated by nipalsPCA.m or plsCV.m

components a vector containing desired components: at least 1

component, but no more than 3. eg: [1], [1 4], or [1 2 3].

Y Optional: if response vector Y is known, use it to color the scores plot

VisLoadings1D(X,loadings,features,range)

Correlation/covariance plot of spectra modeled by loadings or

correlations.

Arguments:

X (N x P) Data matrix of spectra

loadings (1 x P) Vector of coefficients from PLS betas, PCA loadings, or correlations

features (1 x P) Vector of features (chemical shifts or m/z)

range Optional: range of coefficients , default [min max]

[corr,covarX]=VisLoadings2D(X,loadings,ppm1,ppm2,disp,label)

Plots correlation/covariance projection of 2D NMR spectra using a vector of loadings from PCA, PLS, or a correlation vector from univariate regression.

Arguments:

X Data matrix of spectra

loadings Vector of loadings or correlations from binmat or full spectra- eg PCA.loadings

ppm1 Chemical shift vector of F2

ppm2 Chemical shift vector of F1

disp string specifying 'contour' plot or 'fast' plot

label Optional spectral segmentation matrix if using binning.

Length of loadings vector must equal number of segments.

Return Values:

corr Full spectral size correlation matrix

covarX Full size back-projected covariance matrix.

manhattan(X,Y,features,p,sigs,correction)

Manhattan plot of p values calculated by MWAS.m for all features, and

boxplots of all significant features by class.

Arguments:

X Data matrix of spectra

Y Vector of classes

features Vector of features (chemical shifts or m/z)

p P values

sigs Optional: logical vector of significant features

correction Optional: Either Bonferroni, 'bonferroni', or Sidak,

'sidak'. Default is 'bonferroni'

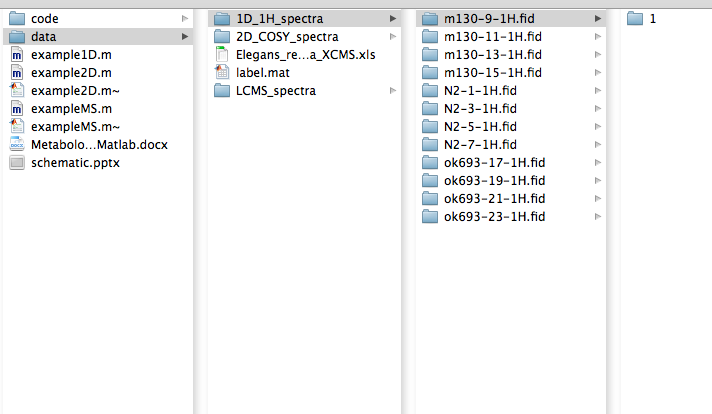
Metabolomics Toolbox

Section 1:

1D NMR Processing Guide

To start: A set of 1D NMR spectra collected and saved in Bruker 1r format. You should process your data to your liking using XWinNMR or Topspin, as these programs output the 1r files you will be using in Matlab. This toolbox does not currently support Varian data, though only a loading script would be necessary to make the toolbox Varian compatible. Here, we have 1D 1H NMR spectra from three strains of *C. elegans*- wild-type (N2) and two mutant strains (*ok693* and *m130*) with four replicates each.

This data can be found under /data/1D\_1H\_spectra in the toolbox folder.



Now open Matlab and set your working directory to the /code directory or set your path to include it.

Loading spectra: If all of our spectra were in one directory and their names are systematic, it would be possible to easily load them all at once. However, because there is an additional directory level between the sample IDs (ie *m130*-9-1H.fid) and the experimental data (ie 1), paths to data must be specified.

path\_1D{1}='../data/1D\_1H\_spectra/N2-1-1H.fid/1';

path\_1D{2}='../data/1D\_1H\_spectra/N2-3-1H.fid/1';

path\_1D{3}='../data/1D\_1H\_spectra/N2-5-1H.fid/1';

path\_1D{4}='../data/1D\_1H\_spectra/N2-7-1H.fid/1';

path\_1D{5}='../data/1D\_1H\_spectra/m130-9-1H.fid/1';

path\_1D{6}='../data/1D\_1H\_spectra/m130-11-1H.fid/1';

path\_1D{7}='../data/1D\_1H\_spectra/m130-13-1H.fid/1';

path\_1D{8}='../data/1D\_1H\_spectra/m130-15-1H.fid/1';

path\_1D{9}='../data/1D\_1H\_spectra/ok693-17-1H.fid/1';

path\_1D{10}='../data/1D\_1H\_spectra/ok693-19-1H.fid/1';

path\_1D{11}='../data/1D\_1H\_spectra/ok693-21-1H.fid/1';

path\_1D{12}='../data/1D\_1H\_spectra/ok693-23-1H.fid/1';

Now load the spectra.

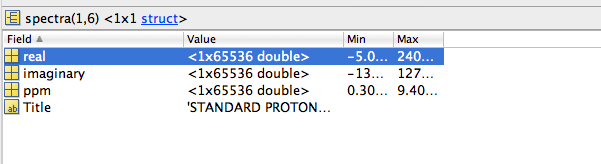
spectra=Load1D(path\_1D,'bruker');

Also, add a Y vector with class membership variables for each spectrum.

Y=[0 0 0 0 1 1 1 1 1 1 1 1]';

Y2=[1 1 1 1 2 2 2 2 3 3 3 3]';

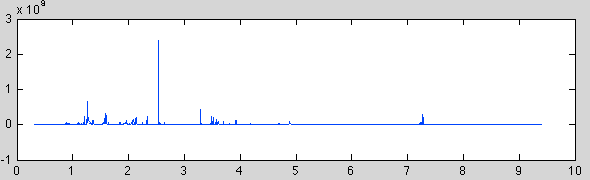
Here, ‘Y’ differentiates between WT and mutant strains, and Y2 specifies strain identities. They will have different uses in the analysis and visualization of this data.



Load1D.m creates a structure called “spectra” with an array for each of the NMR spectra loaded. Here, we are looking at spectra(6) in the Matlab Variable Editor. You can see this by double clicking on spectra in the Workspace and then double clicking on the first array.

Let’s look at this spectrum to make sure it looks okay.

figure, plot(spectra(6).ppm,spectra(6).real)



Notice that Matlab does not automatically obey the convention of orienting the chemical shift scale as downfield going left rather than right. To correct this, enter

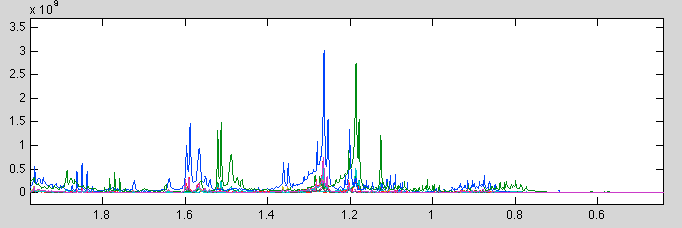
set(gca,'XDir','rev')

Okay, now we are ready to put all the spectra into a single matrix and perform some critical preprocessing steps.

[X,ppm,XTitles]=Setup1D(spectra);

figure, plot(ppm,X)

set(gca,'XDir','rev')



Notice that there seems to be a referencing problem with the green spectrum. Some normalization issues are also apparent, but we will address those later on. Let’s fix the referencing issue quickly.

spectra(9).ppm=spectra(9).ppm+0.077;

[X,ppm,XTitles]=Setup1D(spectra);

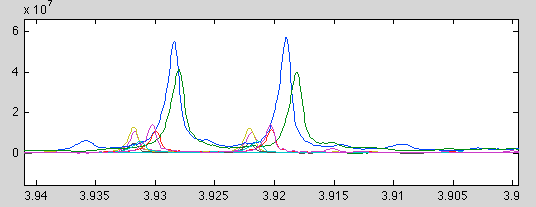
You can plot the spectra again to check that the referencing issue is resolved. It’s good practice to take a look at the stackplot of the data following import to check that everything looks right before any further processing or analysis. This is also a good opportunity to check the signal alignment, and identify any extraneous signals, such as solvent signals, to remove.

To remove solvent (methanol-d4, here) signal, use the following function

XR=remove\_region(X,ppm,4.75,4.95);

XR=remove\_region(XR,ppm,3.275,3.295);

Aligment of peaks actually looks pretty decent in the unaligned data, but alignment for 1D NMR is normally pretty important given the variation in chemical shifts relative to line widths. For example, this region.



To align this peak, either star alignment or guided alignment can be used. Star alignment is simpler and faster, so let’s try that.

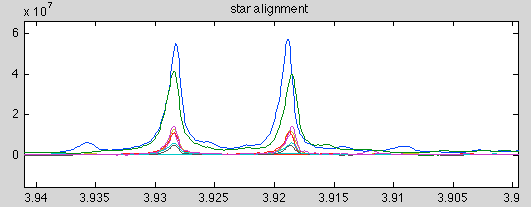
XAL=star\_align1D(XR,ppm,'mean','PAFFT');

figure, plot(ppm,XAL)

set(gca,'XDir','reverse')

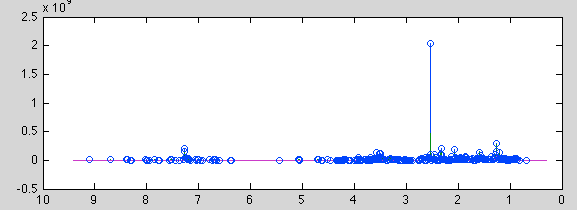
title('star alignment')

Here, we are using the PAFFT package – included in the toolbox is the code for ICOSHIFT, PAFFT/RAFFT, and CCOW. Star alignments align all spectra to a reference- we are using the mean of the spectral data as a reference. Mean, median, and std usually work better as references than using an individual spectrum.



We’re almost done with the preprocessing tools specific to 1D 1H NMR. The goal of preprocessing is to get a 2D (N x P) matrix of metabolite/spectral features where N is the number of samples (observations) and P is the number of features (variables). A matrix of full-resolution 1D spectra works. Reducing spectral width by cropping spectra is useful if there is a significant amount of the spectral domain with no signal present, as datasets with high percentages of noise-dominated variables present issues with scaling and normalization to come later. In some cases, more accurate classification can be achieved by removing noise variables. To allow that, the toolbox includes a peak identification script which will identify peak maxima and create a matrix of intensities at maxima.

[peakmatrix,shifts]=Peakpick1D(XAL,ppm,'std',.7);



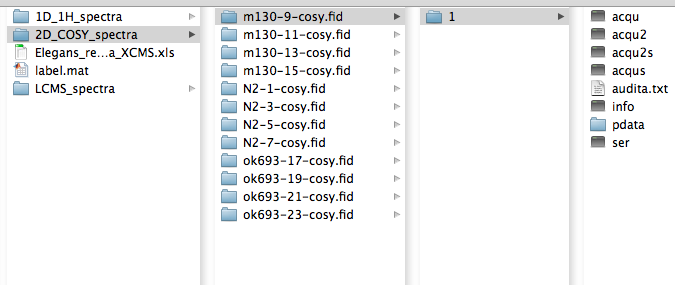
Here, blue circles show the peak maxima identified by the Peakpick1D function. Check to see that the relevant signals have been identified. If not, the threshold for detection can be increased or decreased. This will output an N x P matrix of intensities ‘peakmatrix’ as well as a 1 x P vector ‘shifts’ containing the chemical shifts of each detected maximum. This is optional, but in my opinion it is worth doing to check against the full resolution data when doing analysis (later).

The Metabolomics Toolbox is designed to be modular and not to replicate functionality that can be applied to 1D NMR, 2D NMR, and MS data. If you are processing 1D data alone, skip to section 4 on dataset stabilization.

Metabolomics Toolbox Section 2:

2D NMR Processing Guide

To start: A set of 2D NMR spectra collected and saved in Bruker 2rr format. You should process your data to your liking using XWinNMR or Topspin, as these programs output the 2rr files you will be using in Matlab. This toolbox does not currently support Varian data, though only a loading script would be necessary to make the toolbox Varian compatible. Here, we have 2D 1H-1H COSY spectra from three strains of *C. elegans*- wild-type (N2) and two mutant strains (*ok693* and *m130*) with four replicates each.



Now open Matlab and set your working directory to the /code directory or set your path to include it.

Loading spectra: If all of our spectra were in one directory and their names are systematic, it would be possible to easily load them all at once. However, because there is an additional directory level between the sample IDs (ie m130-9-cosy.fid) and the experimental data (ie 1), paths to data must be specified.

path\_2D{1}='../data/2D\_COSY\_spectra/N2-1-cosy.fid/1';

path\_2D{2}='../data/2D\_COSY\_spectra/N2-3-cosy.fid/1';

path\_2D{3}='../data/2D\_COSY\_spectra/N2-5-cosy.fid/1';

path\_2D{4}='../data/2D\_COSY\_spectra/N2-7-cosy.fid/1';

path\_2D{5}='../data/2D\_COSY\_spectra/m130-9-cosy.fid/1';

path\_2D{6}='../data/2D\_COSY\_spectra/m130-11-cosy.fid/1';

path\_2D{7}='../data/2D\_COSY\_spectra/m130-13-cosy.fid/1';

path\_2D{8}='../data/2D\_COSY\_spectra/m130-15-cosy.fid/1';

path\_2D{9}='../data/2D\_COSY\_spectra/ok693-17-cosy.fid/1';

path\_2D{10}='../data/2D\_COSY\_spectra/ok693-19-cosy.fid/1';

path\_2D{11}='../data/2D\_COSY\_spectra/ok693-21-cosy.fid/1';

path\_2D{12}='../data/2D\_COSY\_spectra/ok693-23-cosy.fid/1';

Add some Y vectors with class membership variables for each spectrum.

Y=[0 0 0 0 1 1 1 1 1 1 1 1]';

Y2=[1 1 1 1 2 2 2 2 3 3 3 3]';

Load the data.

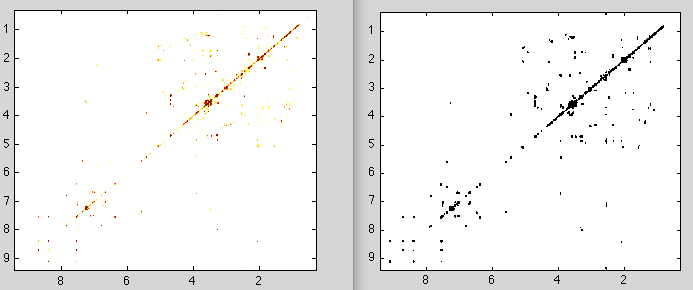
spectra=Load2D(path\_2D);

Like Load1D.m, Load2D.m creates a structure called “spectra” with an array for each of the NMR spectra loaded. Let’s take a look at the first spectrum to check that the import worked correctly. 2D data can be plotted either as a fast plot or as a contour plot. While fast plots are, well, faster, and do a better job with phase-sensitive data (though I recommend magnitude mode processing for the toolbox) contour plots tend to be more visually appealing, are stackable, and are the normal mode of representing 2D NMR data. Lets do both using the vis2D command.

vis2D(spectra(1),'f')

vis2D(spectra(1),'c',20)

Here, we’ll use both fast mode (‘f’) and contour mode (‘c’).



These are the two display modes for 2D spectra. Contour plotting can be computationally demanding with high numbers of contours, so check the threshold (here, 20).

You can also change the fast plot colormap interactively in the Matlab Figure window. Go to ‘Edit’, ‘Colormap’, and change the value for color data min and max (though you should keep their absolute values the same) and slide the color arrows to interactively change the coloring of the fast plot. You can also double click on the color arrows themselves to change the color completely- say from red to green.

Now let’s make a 3D matrix of these 2D spectra.

Setup2D(spectra);

Setup2D outputs a number of variables into the workspace- X, the 3D matrix of spectra, XNoise, the calculated local noise matrices for each spectra, and ppm1 and ppm2 indices for each dimension.

To check the data, do a stackplot. This is useful for preliminary data exploration and to check if preprocessing steps such as alignment are necessary.

stackplot(X,XNoise,ppm1,ppm2,Y2)

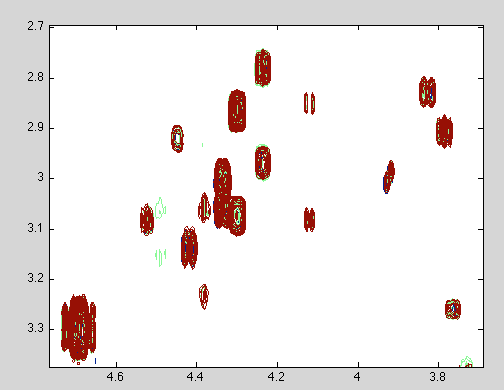
Stackplots can be slow if too many contours are plotted- manually specifying thresholds can help if this is the case. Automatic threshold estimation generally works fairly well, however, and should be tried first. If plotting takes more than ten minutes, try the stackplot function for a subset of spectra.

stackplot(X(:,:,1),XNoise(:,:,1),,ppm1,ppm2,Y2,50);

This specifies a threshold of 50 for spectrum 1 in matrix X. You can then save this in a vector, threshvect or if you figure out all the thresholds and input them at once

stackplot(X,XNoise,ppm1,ppm2,Y,threshvect);

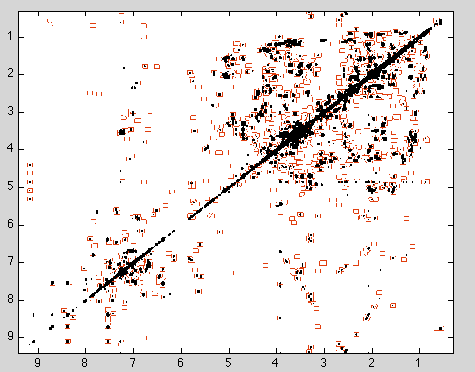
The inclusion the class labels will color the spectra by class.



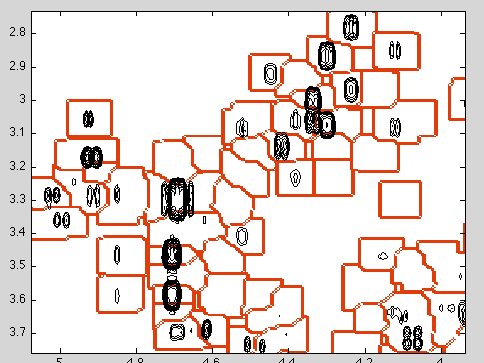
The alignment here is pretty spot-on. Alignment is typically less critical for 2D NMR data than for 1D 1H data as the lower resolution means that chemical shift variation is smaller compared to peak widths than for 1D data. As with 1Ds, we can do either guided or star alignment, but first we need to segment the spectra to identify peak regions. This is the critical step in preprocessing 2D NMR data, as both alignment and binning depend on accurate segmentation.

label=segment2D(X,XNoise,ppm1,ppm2)

This will apply default settings of S/N threshold of 4 (for homonuclear, 10 for heteronuclear), with an allowed distance between local maxima in the same crosspeak of 0.035 ppm (for homonuclear, 0.025 ppm 1H and 0.3 ppm 13C for HSQCs). Segment2D will output a visualization of the detected segments plotted on the mean spectrum.



Zoom in to check the segmentation. As is evident from below, spectral segments are not regular, but are adaptive based on the bounding region of the crosspeaks. Detection regions can be modified by altering the maximum allowed distance between maxima in the same crosspeak. These segments form the basis of alignment and binning- alignment shifts signals within the segments, while binning integrates the regions for all spectra.



Segmentation can take quite a while, especially without Matlab’s Image Processing Toolbox. For the demo data, I’ve saved the label matrix calculated with default settings in /data/label.mat. Load this with

load('../data/label.mat')

As mentioned, alignment is not really necessary for these spectra, but if alignment is desired, HATS.m and star\_align2D.m can be used for guided and star alignment, respectively. See the example2D.m script for more details and check above for syntax.

Now bin the data using the label matrix.

binmat=bin2D(X,label);

While pattern recognition can be applied to full-resolution 2D NMR using unfolding/refolding, binning tends to produce better prediction.

The Metabolomics Toolbox is designed to be modular and not to replicate functionality that can be applied to 1D NMR, 2D NMR, and MS data. If you are processing 2D data alone, skip to section 4 on dataset stabilization.

Metabolomics Toolbox Section 3:

MS Processing Guide

While MS data can be preprocessed in Matlab, XMCS (R package) is several years ahead in terms of many of the preprocessing steps including feature detection, matching, and alignment and has an active development community. For that reason, I recommend using XCMS in R to preprocess LC-MS data, and to then save the XCMS output file in .XLS format for import into Matlab.

I’m not going to go in-depth into using XCMS here, but I use R-Studio (http://rstudio.org/) . Installation instructions for XCMS can be found at (<http://www.bioconductor.org/packages/2.9/bioc/html/xcms.html>) and the package and all dependencies are available via CRAN.

I tend to simply replace the CDF data in the ‘faahKO’ package with my CDF data and run the following code:

library(xcms)

library(faahKO)

cdfpath <- system.file("cdf", package = "faahKO")

cdffiles <- list.files(cdfpath, recursive = TRUE, full.names = TRUE)

xset <- xcmsSet(cdffiles)

xset <- group(xset)

xset2 <- retcor(xset, family = "symmetric", plottype = "mdevden")

xset2 <- group(xset2, bw = 10)

xset3 <- fillPeaks(xset2)

reporttab <- diffreport(xset3, "WT", "KO", "example", 10, metlin = 0.15, h=480, w=640)

write.csv(reporttab, file = "Elegans\_report\_data\_XCMS.csv", row.names = FALSE)

XCMS provides many more options, of course, and these are just default settings. Save the output file, here ‘Elegans\_report\_data\_XCMS.csv’ in XLS format. This output file can be found in /data/

To import into Matlab, use the following code:

importXCMS('../data/Elegans\_report\_data\_XCMS.xls')

XCMS uses a different import order, so check this in the XLS file. To match up the MS data with the NMR data, use

reorder=[9 10 11 12 4 1 2 3 5 6 7 8];

msmatrix=msmatrix(reorder,:);

Metabolomics Toolbox Section 4:

Data Normalization/Stabilization

The Metabolomics Toolbox is designed to be modular and not to replicate functionality that can be applied to 1D NMR, 2D NMR, and MS data. For this reason, normalization and variance stabilization techniques are applicable for any 2D (N x P) matrix – this can be X/peakmatrix for 1D NMR, binmat for 2D NMR, or msmatrix for MS data. I’ll use the 1D NMR data here because it will be easy to visualize results, but these functions are equally applicable to any data matrix. Refer to example2D and exampleMS for more details.

The purpose of normalization is to counter for the intensity multiplier effects- normally this means concentration, but can include receiver gain, ion saturation, etc- any effect that will modify approximately all signal intensities by a (linear) multiplier.

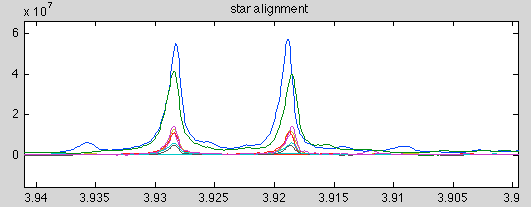
The purpose of stabilization/scaling is to balance the variance of major peaks vs. minor peaks. Un-scaled datasets are dominated by the variance of intense peaks, and pattern recognition will highlight major peaks with potentially little relevant biological variance at the expense of minor, but significant peaks.

The functionality included in this toolbox is meant to not only correct for these effects, but also to visualize, explore, and help understand what corrections are necessary.

Let’s start by checking the normalization of our full-resolution 1D dataset XAL.

figure, plot(ppm,XAL)

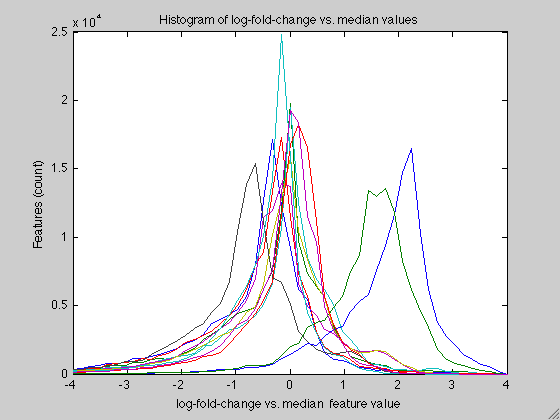
set(gca,'XDir','reverse')

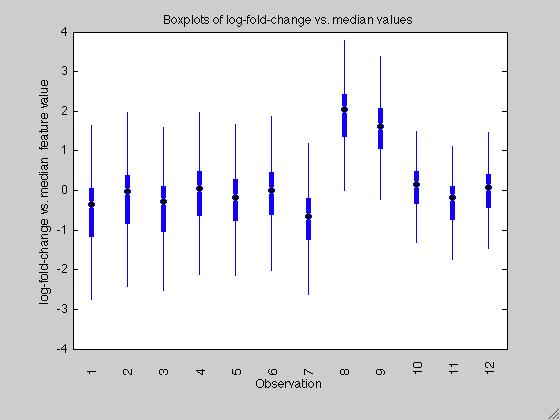


As before, we notice that two spectra seem to have higher intensities across the ppm domain – the blue spectrum and the green spectrum.

normcheck(XAL)

Normcheck plots histograms and barplots of log-fold change vs. median for all spectra. Notice that these spectra are right-shifted with high fold-changes for nearly all datapoints. This distribution indicates a normalization issue.

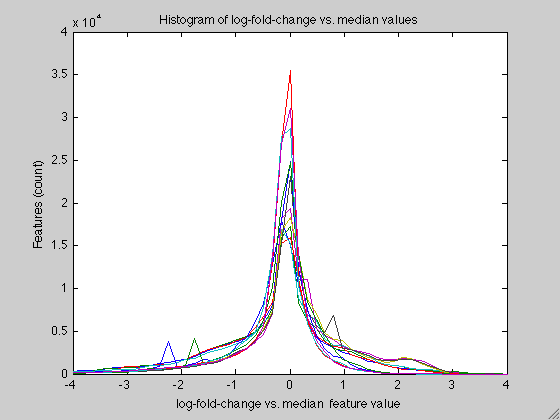




Normalize.m contains the functionality for normalizing this data. In cases where a large (>50%) percentage of datapoints are linearly increased/decreased in intensity, probabilistic quotient normalization works quite well. However, in other cases where normalization to an internal reference, such as TSP or CRN is desired, spectra can be normalized to peak intensities or integrals. Total area normalization is also supported, but tends to be less useful.

XALN=normalize(XAL,'PQN');

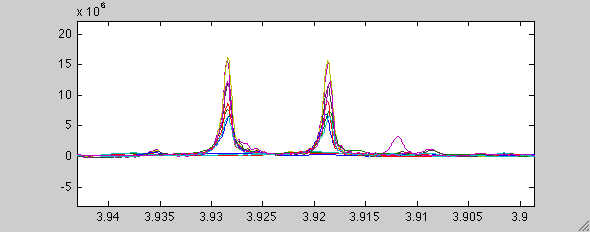
normcheck(XAL)



Notice that LFC distributions are now centered at 0.

figure, plot(ppm,XALN)

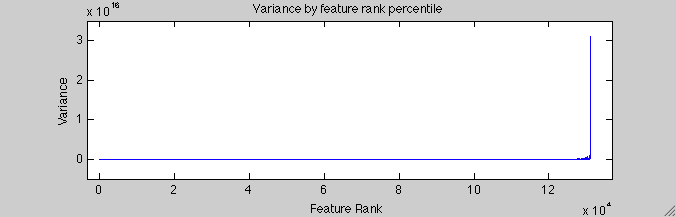
set(gca,'XDir','rev')

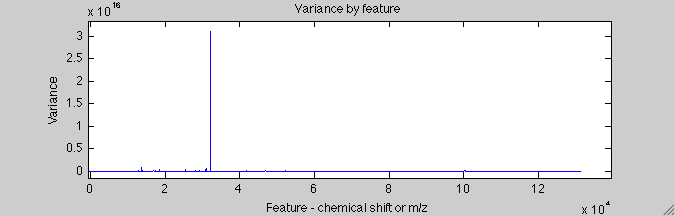


This resolves previously mentioned with the non-normalized data. Also, potential class differences in this signal are now more notable.

Let’s now check the scaling of this data.

varcheck(XALN)



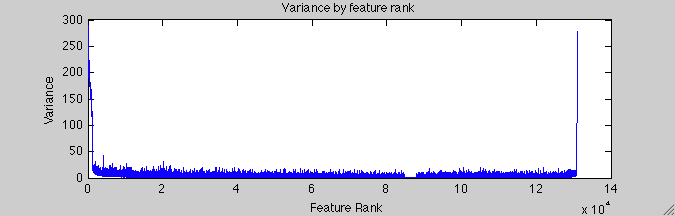


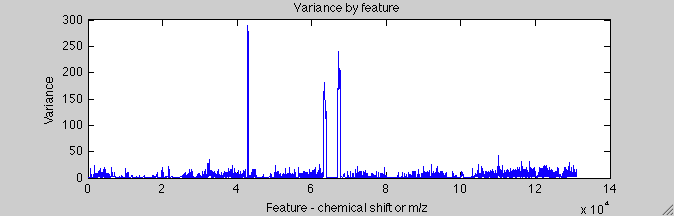
These two plots show signal variance by rank (top) and chemical shift (bottom). They indicate that a small number of intense peaks (well, one singlet, really) dominate the variance of the un-scaled dataset.

Now let’s try log scaling. ‘log’ and ‘logoff’ both use log-2 transformation.

XALSN=scale(XALN,'log');

varcheck(XALSN)

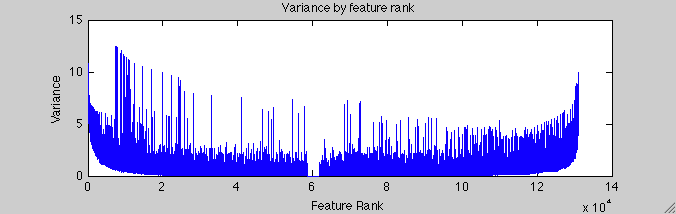


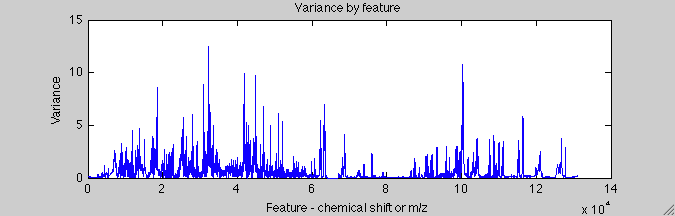


This certainly reduces the relative variance of the intense signals, but also up-scales the noise quite a bit. Variance plotted by chemical shift shows that most peaks blend in with the variance of the noise. Let’s try adding an offset. Here, the automatic offset is the median of all values greater than zero. An offset approaching the intensity of the highest peaks has no scaling effect, while an offset approaching zero is the same as not applying an offset.

XALSN=scale(XALN,'logoff');

varcheck(XALSN)





Note that the variance is balanced across the rank space, and that peaks tend to comprise the variance. This is a desired outcome.

Varcheck also outputs a histogram of variance- under-scaled data that tends to be right-tailed, while over-scaled data is left-tailed.

Metabolomics Toolbox Section 5:

Data Analysis / Pattern Recognition

The Metabolomics Toolbox is designed to be modular and not to replicate functionality that can be applied to 1D NMR, 2D NMR, and MS data. For this reason, analysis methods shown here are applicable for any 2D (N x P) matrix – this can be X/peakmatrix for 1D NMR, binmat for 2D NMR, or msmatrix for MS data. I’ll use the 2D NMR data here, but these functions are equally applicable to any data matrix. Refer to example1D and exampleMS for more details.

For data analysis using the 2D NMR data, we’ll need our 3D matrices X and XNoise, along with ppm1 and ppm2. We’ve also applied normalization and stabilization to the matrix of segment integrals ‘binmat’, to get ‘binmatSN’.

**A note to users**: much of the functionality in this section of the toolbox depends on functionality in the Matlab Statistics Toolbox. To check if you have the statistics toolbox installed, enter the code below in the command window:

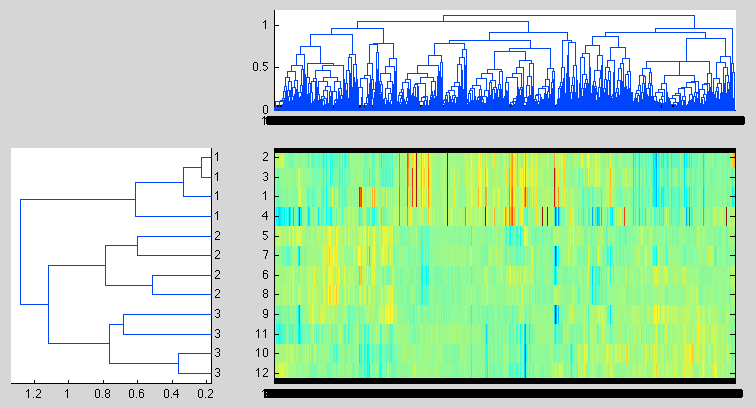
ver('stats')

The output should show the version number of your statistics toolbox. If the toolbox is not installed, some of the following functions will not work properly.

Let’s start out with a two-way hierarchical cluster analysis.

[sample\_order,variable\_order]=two\_way\_cluster(binmatSN,'weighted','spearman',Y2);

Here, we’re using weighted agglomerative clustering with a Euclidean distance metric. Two\_way\_cluster.m calls the Matlab function linkage.m, so the options for clustering methods and metrics from linkage.m can be used for two\_way\_cluster.m as well.

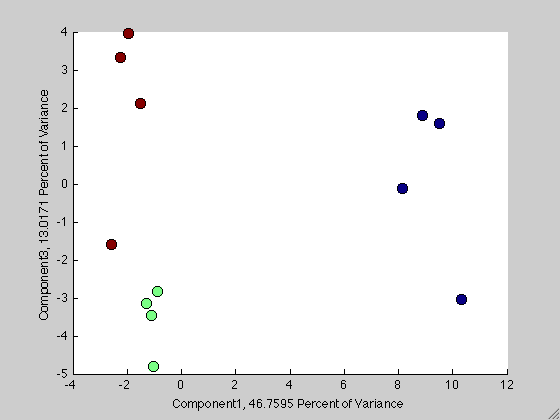


We can see from these dendrograms that the sample classes cluster well. The heatmap also shows red (+ LFC) features for the Class 1 (wild-type) samples. Because this data was processed with LFC-vs-median, the smaller number of wild-type samples mean than the median values will be shifted towards mutant strain levels giving the wild-type features larger magnitude LFC values. This will be evident in pattern recognition as well.

PCA=nipalsPCA(binmatSN,5);

VisScores(binmatSN,PCA,[1 3],Y2);

nipalsPCA.m uses the NIPALS algorithm to identify the first N principal components of the data. Here, we have instructed it to identify scores and loadings for the first 5 principal components of X. PCA identifies linear combinations of variables which maximize the variance in the dataset.

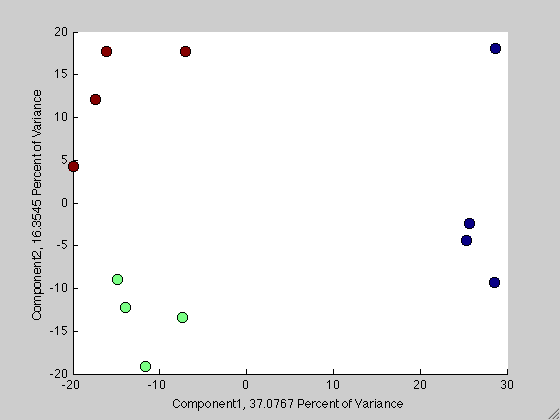


Notice that the wild-type (blue) scores are ~ 8 – 10 on PC1, while the mutant scores are ~ -1 – -2. This imbalance will show up in loadings as well. There are two potential ways to get around this issue- the first is to use log-fold-change vs. mean rather than median, or to use autoscaling, which is less sensitive to these differences.

binmatSN2=scale(binmatN,'auto');

PCA=nipalsPCA(binmatSN2,5);

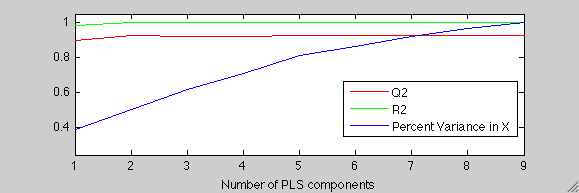
VisScores(binmatSN2,PCA,[1 2],Y2);



There is still a difference in score magnitude, but it is less substantial. The greater separation of the mutant strains by autoscaling also suggests that smaller intensity peaks make a greater contribution to the difference between the mutant strains than between the wild-type and mutants.

We’ll take a look at the loadings in a bit, but let’s first do a PLS so we can compare the results of PCA and PLS.

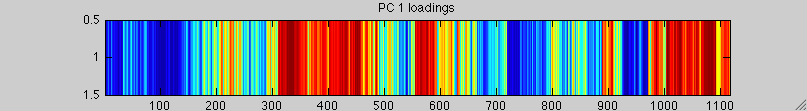
PLS=plsPV(Y,binmatN,5,'da',10,'auto');

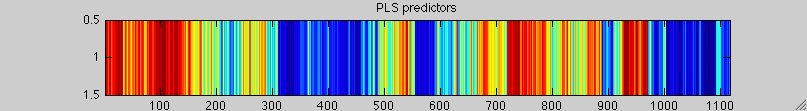


plsPV requires input to select the optimal number of PLS components to avoid overfitting. Here, 5-fold validation indicates that with a single component R2 is very near 1 and Q2 is approximately 0.9. The Q2 values is most important, as this indicates goodness of fit for the cross-validation samples. These results make sense given the PCA results- there is strong separation of WT and KO strains in the first component, so it is little surprise that only a single component is required for highly accurate predictions in PLS. So input ‘1’ into the Matlab command line.

The reason for passing the unscaled binmatN matrix with the ‘auto’ scaling option is to avoid bias during validation- plsPV will perform scaling during the validation step for the training and test data.

Before doing a back-projection, lets compare the PC1 loadings with the predictor coefficients from PLS. We can use the order from clustering to match these loadings up directly with features in the heatmap as well.





figure, imagesc(PCA.loadings(1,variable\_order))

caxis([-1\*max(max(abs(PCA.loadings(1,:)))) 1\*max(max(abs(PCA.loadings(1,:))))])

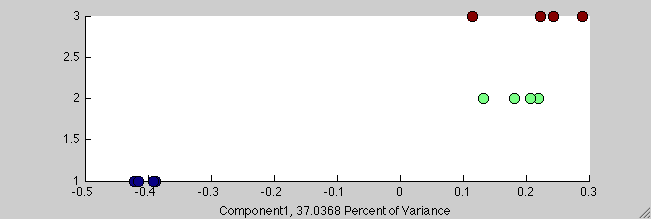
predictors=PLS.betas(2:end);

figure, imagesc(predictors(variable\_order))

caxis([-1\*max(abs(predictors)) 1\*max(abs(predictors))])

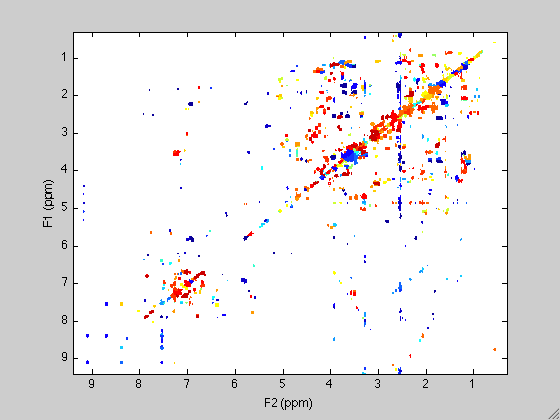
Unsurprisingly, PC 1 and PLS are picking up the same features, with almost the same weights for each. In fact, the R2 between the two is .99

One difference between the two is the phase- why is this? Notice that PC 1 has the WT sample with positive scores, and the KO strains with negative scores. This means that high-WT features will have positive loading coefficients, while high-KO features have negative loadings. The Y vector we used for PLS had a ‘0’ value for WT and a ‘1’ value for KO. Therefore, the pattern will be flipped in PLS. We can see this by plotting PLS scores for component 1.



Now let’s do a back-projection of the PLS predictors to take a look at the crosspeaks flagged as differential.

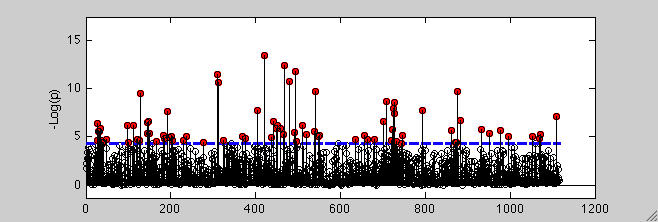
VisLoadings2D(X,predictors,ppm1,ppm2,'c',label);



Here, matching the PLS scores and the Y vector we used for discriminant analysis, high-WT features will have negative loading coefficients (blue), while high-KO features have positive loadings (red). If the opposite is desired, Y=1 can be used for WT and Y=0 for KO. It looks like there are a lot of highly variable features between the WT and KO classes. A quick significance test will confirm that.

[p,sigs]=MWAS(binmatSN,Y,'bonferroni');

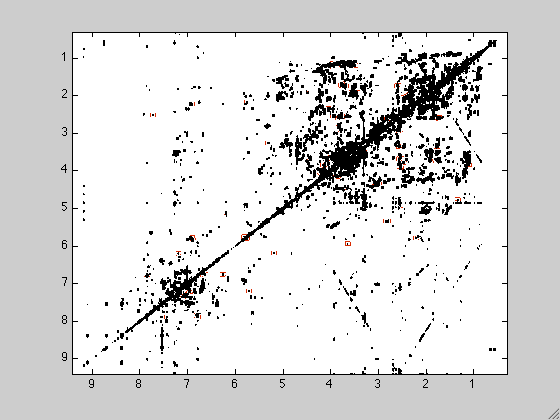
manhattan(binmatSN,Y,1:size(binmat,2),p,sigs,'bonferroni')



In this Manhattan plot, the significance threshold calculated by Bonferroni multiple hypothesis correction is plotted in blue, while features with p-values above this threshold are highlighted in red. There are quite a few, though investigation of the most significant features is likely the best place to start interpreting this data.

To display all crosspeaks identified as significant, use the displaypeak2D.m function.

displaypeak2D(X,XNoise,label,find(sigs==1),ppm1,ppm2)



Alternately, individual indices can be passed to the display2D function. For example, to display the region corresponding to binmat(:,1), use the following syntax:

displaypeak2D(X,XNoise,label,1,ppm1,ppm2)

That wraps up the documentation for the Metabolomics Toolbox.