Report Title

Development and Validation of a Method for Individualization of Middle Petroleum Distillates and Kerosene Ignitable Liquids

Award Number NIJ 2008-DN-BX-K146

Author /Principal Investigator	J. Graham Rankin, PhD, Professor Forensic Science Program Marshall University 1401 Forensic Science Dr. Huntington, WV 25701 Rankinj@marshall.edu 304-690-4377
Co-Principal Investigator:	Peter deB. Harrington, PhD, Professor Department of Chemistry and Biochemistry Clippinger Laboratories Ohio University, Athens Athens, OH 45701 Peter.Harrington@Ohio.edu 740-517-8458
Administrative POC:	William Byrd Marshall University Research Corporation 401 11th Street Huntington, WV 25701 byrdw@marshall.edu 304-696-6324
Financial POC:	Jan Weece Forensic Science Center Marshall University 1401 Forensic Science Dr. Huntington, WV 25701 Weece@marshall.edu 304-691-8957

Abstract

This project applied target compound analysis from high resolution Gas Chromatography/Mass Spectrometry (GCMS) utilized for gasoline individualization to Medium Petroleum Distillates (MPD) and kerosene which may be used as accelerants in arson cases. A major goal of this project was to establish analytical tools that provide a statistical evaluation for forensic comparison between ignitable liquid residues in fire debris and ignitable liquids in the possession of a suspect. Classifications based on two-way profile and target component ratios were compared. For kerosene, the Projected Difference Resolution (PDR) mapping technique was applied to measure the quantitative differences among the Ignitable Liquid (IL) samples by their GCMS profiles. Fuzzy Rule-building Expert Systems (FuRESs) were applied to classify individual ILs. The FuRES models yielded correct classification rates greater than 90% for discriminating between samples. PDR mapping, a new method for characterizing complex data sets was consistent with the FuRES classification result. For MPD, the Kendall's-tau metric for 'association/no-association' was utilized to demonstrate 'association/no-association' true positive rates in excess of 95% with false positive rates ~5%.

Table of Contents

Title Page ……. 1
Abstract ……. 2
Executive summary…… 4
I. Introduction……... 9
a. Statement of Problem …… 9
b. Statement of rationale for the research ..…… 9
II. Methods …… 11
a. Sample Source and collection ……... 11
i. Kerosene …… 11
ii. Middle Petroleum Distillates …… 13
b. GCMS Instrument parameters ……. 13
c. Comparisons …… 14
d. Statistical methods …… 14
e. Evaporation study ……..…… 18
f. Burn Tests ……..…. 19
III. Results …… 19
a. Kerosene Analysis …… 19
i. Baseline correction ……. 20
ii. PDR mapping ……. 21
iii. Pattern Classification ……..….. 22
b. Middle Petroleum Distillates ……. 24
i. Chemometric Analysis of MPD target compound peak area ratios 25
c. Evaporation Study ……..…….. 28
d. Burn Test ……... 30
IV. Conclusions …… 30
a. *Discussion of findings* …….. 30
b. *Implications for policy and practice* …… 31
c. Implications for further research …… 31
V. Acknowledgements …… 32
VI. References ……..…… 32
VII. Dissemination of Research Findings ……..… 34
a. Published Papers and Proceedings ……. 34
b. Dissertation ……. 35
c. Presentations at National and Regional Meetings …… 35
d. Database …… 35
VIII. Appendix 1: List of Kerosene Samples ……..36
IX. Appendix 2: List of Middle Petroleum Distillate Samples ……39
X. Appendix 3: MATLAB routines utilized for Kerosene Data ……43
XI. Appendix 4: R Scripts utilized for MPD Data ……..…74

Executive Summary

Description of the Problem

Gasoline and kerosene are the two most common ignitable liquids used as accelerants in arson cases. These products are readily available and can be bought in quantity without arousing suspicion. According to the American Society for Testing and Materials (ASTM) E1618 method, products classified as middle petroleum distillates (MPD) are also commonly used ignitable liquids. MPD are readily available in the form of charcoal lighters, paint thinners, and industrial solvents making them other readily available choices for arsonists.

Fire debris analysis (FDA) is the science that involves examination of fire debris samples performed to detect and identify ignitable liquid residues (ILR). Using the E1618 classification scheme can readily identify the ignitable liquid residue as a gasoline, MPD, or kerosene; however, comparing two samples (i.e. residue from fire debris to a liquid sample in the suspect's possession) to determine if they are from the same source is much more problematic. The field of fire investigation is very subjective because of the high degree of visual interpretation needed. Recently a number of different statistical based methods have been published. No single method for the statistical comparison of ignitable liquids has proven to be the best in all cases involving fire debris analysis. Although classification of the type of liquid used to create a fire is important, distinguishing within a specific class is more necessary to the world of fire investigation. Therefore, it is important, however complicated it is, to be able to compare two or more samples in a case, to determine if the ignitable liquid residues share a common source. In addition, determination of the precision and error rates for the comparison of IL samples is also important for the evidence to have legal standing in many Daubert states.

The recent National Academy of Science report recommends that pattern recognition techniques (of which ignitable liquid residue analysis is one) have established error rates to meet Daubert rules of evidence in court.

Purpose, Goals and Objectives:

In this study, high resolution GCMS and target compound analysis was used to develop a valid method in order to statistically differentiate between different kerosene and MPD samples. Kerosene and MPD, unlike gasoline, are simple distillation products from crude oil and should be strongly related to the petroleum from which it was distilled. The relative concentrations of the key components in kerosene and MPD from a refinery will often change daily because of the variety of sources that distribute crude oil. This variation in concentration of the key compounds can provide sufficient variability for comparison between individual samples. Kerosene was evaluated from a single refinery to first establish a target compound list. Applying target compound ratio analysis to all of the kerosene samples established a method for differentiation. For MPD, a selection of commercial products (paint thinners and charcoal lighters) was used for establishing the corresponding peak ratios along with samples from the Ignitable Liquid Collection maintained by the Technical Working Group for Fire and Explosives (TWGFEX). The key objectives can be summarized as:

1. Collecting a large set of kerosene and MPD samples and analyzing them by high resolution GCMS.
2. Selecting a number of candidate target compounds common to most samples which elute in pairs relatively close in elution time and have repeatable peak area ratios (ideally within 5% RSD) within samples, but variable between samples of different origin.
3. Using multivariate statistical methods, determine those candidate peak ratios which are best for discriminating between samples for kerosene and MPD.

4. Determine the “false positive rate” when comparing all the samples in the data set.
5. Determine the robustness of the method for evaporated and simulated burned samples.

Research Design and Methods

Kerosene samples were obtained from the Marathon-Ashland refinery in Catlettsburg, Kentucky near Marshall University. The quality control laboratory at the refinery receives samples from all Marathon refineries and distribution terminals in a nine state region. Additional samples were purchased locally from service stations as well as home improvement stores. The original goal of obtaining samples from a wider region proved impractical due to Department of Transportation restrictions requiring ‘certified’ shippers in each state. For MPD samples, a number of commercial products which frequently contain MPD (charcoal lighters, paint thinners and solvents) were purchased locally and supplemented by samples from the Ignitable Liquid Reference Collection administered by TWGFEX.

GCMS analysis was performed using a 60m polymethylsiloxane column which had been shown to separate gasoline individualization studies. Usually a 30m column is recommended for routine fire debris analysis for ignitable liquid residues for classification by the ASTM E1618 method. The longer column used in this study allows for better separation of closely eluting compounds at the expense of longer analysis time. Integration of peak areas utilized standard methods included in the instrument software as well as routines developed using MATLAB.

Statistical methods used to establish error rates included a number of different forms of multivariate statistics including principal components analysis (PCA), projected difference resolution (PDR) mapping, fuzzy-rule building expert systems (FuRESs), Pearson correlation coefficient, Spearman’s-rho rank correlation coefficient and Kendall’s-tau coefficient.

Results:

A total of 76 kerosene samples were analyzed at least three times each. Of that data set 44 were selected for developing the kerosene model. Thirty-six target compounds were selected and 35 peak area ratios determined. For MPD, 44 commercial samples of charcoal lighters, paint thinners and other solvents were determined by E1618 method to be classified as MPD. These samples along will 111 samples obtained from the ILRC which were classified as MPD were analyzed. Forty-one target compounds were selected resulting in 33 peak area ratios for the MPD samples.

For the kerosene analyses, the peak integration routines in the instrument software did not give consistent results as the MATLAB routines processing the raw data files. Differences in background correction for the increasing baseline from column bleed at higher temperatures may have contributed to this observation. For MPD, which elute well before significant column bleed, like previous studies with gasoline, were not affected and did not need special background correction routines.

For kerosene, the FuRES models yielded correct classification rates greater than 90% for discriminating between samples. PDR mapping, a new method for characterizing complex data sets was consistent with the FuRES classification result. For MPD, the Kendall's-tau metric for 'association/no-association' was utilized to demonstrate 'association/no-association' true positive rates in excess of 95% with false positive rates ~5%.

Evaporation studies, although limited, gave results as expected. Lower boiling point components were lost and those peak area ratios containing those components were affected. Some preliminary studies on the effects of substrate were inconclusive and need to be expanded. Such studies are on-going related to substrate effects on E1618 classification.

Conclusions:

This method shows promise for comparison of ignitable liquids in these two classes as well as gasoline, which has been previously reported. Although the false positive rate (less that 5%) will never approach that of DNA or some other tests, it is more likely that two samples can be excluded as being significantly different. Additional work, especially with a larger data set of kerosene samples, optimization of software routines to analyze data from a variety of vendors' instruments and continued work on the effects of evaporation and substrate are needed.

As with any determination of the presence of an ignitable liquid in fire debris, this does not necessarily mean that the ignitable liquid was used as an accelerant, but may be incidental to the scene.

*Implications for policy and practice***.**

Application of high resolution GCMS analyses for comparison between neat ignitable liquids is possible using pattern matching techniques described in this report with some reasonable degree of statistical validity. A reference database of target compound ratio analyses of gasoline, kerosene and MPD samples such as from the ILRC should allow individual laboratories to utilize these techniques without having to invest considerable time, manpower and expense to recreate the data. Key to the adoption of such a database will be additional studies incorporating different instrumentation and laboratories utilizing a 'round robin' approach.

Implications for further research.

Initial studies in our laboratory have shown that there may be some effects due to substrate and/or evaporation, thus additional studies are needed to test the 'robustness' of this method.

I. Introduction:

Statement of the problem: This research applied target compound analysis that was utilized for gasoline individualization and applied it to medium petroleum distillates (MPD) and kerosene which may be used as accelerants in arson cases. A major goal of this project was to establish analytical tools that provide a statistical evaluation for forensic comparison between ignitable liquid residues in fire debris and ignitable liquids in the possession of a suspect. Establishment of the statistically based error rates is essential in order to meet current and future legal challenges.

Statement of rationale for the research: Gasoline and kerosene are the two most common ignitable liquids used as accelerants in arson cases (1). Ignitable liquids are petroleum based or related products that have certain flammable or combustible properties (2). Products classified as middle petroleum distillates (MPD) according to the American Society for Testing and Materials (ASTM) E1618 method are also commonly used ignitable liquids (3). MPD are readily available in the form of charcoal lighters, paint thinners, and solvents making them another good choice for arson crimes. Although 'arson' is a legal term, a commonly acceptable definition of the word is "a criminal act of deliberately setting fire to a property" which calls for the need of fire investigation (1).

Fire debris analysis (FDA) is the science related to the examination of fire debris samples to detect and identify ignitable liquid residues (ILR) (1). Classification of ignitable liquids according to the E1618 classification scheme can readily identify the ignitable liquid residue as a gasoline, MPD, or kerosene; however, comparing two samples (i.e. residue from fire debris to liquid in the suspect's possession) to determine if they are from the same source is much more problematic. The field of fire investigation is very subjective because of the high degree of visual interpretation needed. No single method for the statistical comparison of ignitable liquids has proven to be the best in all cases involving fire debris analysis. Although classification of

the type of liquid used to create a fire is important, distinguishing within a specific class is often necessary to the world of fire investigation (1). Therefore, however complicated it is to compare two or more samples in a case, it is important to determine if the ignitable liquid residues share a common source. In addition, determination of the precision and error rates for the comparison of IL samples is also important for the evidence to have legal standing in many Daubert states. The recent National Academy of Science report recommends that pattern recognition techniques (of which ignitable liquid residue analysis is one) have established error rates to meet Daubert rules of evidence in court(4,5).

Gas Chromatography Mass Spectrometry (GCMS) is the standard analytical technique used in fire debris analysis. A separation process must occur in order to analyze ignitable liquids. GCMS can be used to produce qualitative and quantitative results because of the composition of ignitable liquids which are composed of many different components based on their production process. When analyzing ignitable liquids produced or distilled in the same manner and classified in the same category, differentiation can be challenging. In order to find differences in similarly classified ignitable liquids, target compound analysis can be used.

Target compound analysis allows the data system to search for specified retention time windows for the mass spectra of specific compounds expected to elute within each window (2). As the target compounds are identified, the data system provides quantitative data containing the peak area of the key components found. Comparison of these key components is done by peak area ratios. Early work by Mann demonstrated the value of using GCMS and calculating peak-to-peak ratios for comparing one gasoline to another (6). Following work done by Mann, Keto (7) and later Dolan and co-workers (8, 9) applied peak ratio analysis to sequential peaks in order to establish a unique identifying profile for each evaluated gasoline sample. The subtle variations in peak area cause variations in peak area ratios resulting in a bigger variation between similarly

classified samples. Although these variations are unique, comparing neat samples to fire debris analysis is not as straightforward.

Fire debris samples are not as easy to analyze as neat samples because of their level of contamination. Contamination, usually the result of pyrolysis of organic materials (wood, carpet, padding, etc.) at the fire scene, adds unwanted peaks to the gas chromatographic pattern (1). These unwanted peaks could cause incorrect visual interpretation. The mass spectrometer's data system is used to "filter out" contaminating species in the chromatogram and produce data that is petroleum distillate related based on its target ions (8).

In this study, GCMS and target compound analysis was used to develop a method like Dolan in order to statistically differentiate between kerosene and MPD samples. Kerosene and MPD, unlike gasoline, are simple distillation products from crude oil and should be strongly related to the petroleum from which it was distilled. The relative concentrations of the key components in kerosene and MPD from a refinery will often change daily because of the variety of sources that distribute crude oil. This variation in concentration of the key compounds can provide sufficient differences for comparison between individual samples. Kerosene was evaluated from a single refinery to first establish a target compound list. Applying target compound ratio analysis to all of the kerosene samples established a method for differentiation. For MPD, a selection of commercial products (paint thinners and charcoal lighters) was used for establishing the corresponding peak ratios.

II. Methods

Sample Source and collection

Kerosene

The kerosene used for this research was collected from the Marathon Ashland Refinery and from commercial sources in Huntington, West Virginia (Figure 1). Samples from the Quality

Assurance lab at Marathon Ashland were received from Marathon refineries and distribution terminals primarily from the Midwestern states. The samples were received over several years representing both seasonal and spatial variation of kerosene. A complete list of samples with collection dates is given in Appendix 1. The kerosene samples from Marathon were received in one-liter glass bottles and were immediately transferred into 125 mL bottles with Teflon®-lined lids to prevent evaporation. Sub-samples of each of the ignitable liquids were refrigerated in 20 mL glass vials. After samples were taken from the refinery bottles, excess kerosene was mixed in a 5 gallon Scepter® gasoline storage cans to be used as a QA/QC kerosene composite sample.

Figure 1: Geographical distribution of 50 kerosene samples collected from 9 states. Refineries or distribution terminals located in some states may distribute to neighboring states.

Middle petroleum distillates

The medium petroleum distillate (MPD) samples used for this experiment were sampled from two primary sets of known liquids. First, 44 MPD samples from our lab's in-house collection of various ignitable liquids were analyzed. These samples consisted primarily of commercial paint thinners and charcoal lighters purchased from home improvement stores in the Huntington, West Virginia area over a period of several years. Identification of these liquid samples was made according to the E1618 guideline. Second, 111 MPD samples from the Ignitable Liquid Reference Collection (ILRC) developed by the ILRC Committee of the Technical Working Group for Fire and Explosives (TWGFEX) were extracted and analyzed (10). These samples were classified as MPD by the ILRC committee. A complete list of samples is given in Appendix 2.

Samples analyzed from the in-house MPD collection were run as neat samples. A 1.5 ml GC vial was filled with the respective MPD and capped. Samples from the ILRC collection were received adsorbed onto small charcoal pieces. Five to seven of these charcoal pieces were placed into a 1.5 ml GC vial and the ignitable liquid residues were desorbed off with pentane. The pentane solvent and extracted ignitable liquid residues were then transferred to a new clean 1.5 ml GC vial and capped.

GCMS Instrument parameters

All analyses were performed on an Agilent 6890N Network GC System coupled to an Agilent 5973 Network Mass Selective Detector. A Varian 60 m DB-1 column with an internal diameter of 250 μm and a 1 μm film thickness was used. The carrier gas was ultra-purity helium for all analyses.

Kerosene: For neat kerosene samples the injection was 0.1 μl with a split flow of 50:1. For diluted samples (E1412 activated charcoal strip passive adsorbent method eluted with carbon

disulfide) the injection was 1 µl with split flow of 30:1. The injector temperature was 250 °C. Oven temperature program: Initial 100°C with 1.0 minute hold time. Linear temperature ramp of 5°C/min to 275°C with a 5.0 minute final hold time.

Middle Petroleum Distillates: For the neat injections of the in-house samples, a split ratio of 50:1 was used. For the ILRC samples and E1412 samples from burn studies, a split ratio of 20:1 was used. The injector temperature was 250 °C. Oven temperature program: Initial 125°C for 1 minute followed by a temperature ramp of 5°C/min to 250°C, hold for 5 minutes.

Comparisons

Target compound analysis was used to identify the key components in both MPD and kerosene using ChemStation software as the output tool for the GCMS data. Parameters were entered into the software program to select the peaks of interest based on retention time and the presence of target ions. A spreadsheet template was developed, similar to that of Dolan and Ritacco (7) for gasoline, and calculated sequential ratios from the averages of the three GCMS injections using Excel©. A relative standard deviation (RSD) less than 5% was used as the criterion for acceptable repeatability among 3 individual injections of each sample (7). Where possible, retention times and mass spectra were confirmed using pure hydrocarbon standards. However, in some cases, no pure standards were available for particular isomers. In those cases, the compounds were identified as "A trimethyl –benzene" or where less certain "Compound A". This follows the practice of Dolan and co-workers (7,8) for gasoline.

Statistical methods

Principal components analysis (PCA) is a multivariate statistical technique which seeks to reduce the dimensionality of large data sets that result from "hyphenated methods" such as GCMS. In such data sets there is often a high degree of correlation between variables (i.e. total

ion intensities at adjacent time slices across a chromatographic peak). PCA fits a sequence of multilinear equations to the variables in the data set to explain the observed variance. Each succeeding equation is fit to the residual variance not explained by the previous one. Ideally a number of strongly correlated variables are collapsed into a few "latent variables". In our data these may be related to actual key compounds or ratios of compounds in the ignitable liquid which vary significantly between samples. Examining the "loadings" (essentially the coefficients of the latent variables) can be useful in selecting key chemical components which can differentiate the ignitable liquids in the data set. (10) The distance in the multidimensional factor space between any two samples is a measure of their chemical difference. Two ignitable liquids which came from a common source (i.e. same station on the same day) should be indistinguishable within some measure of statistical confidence. There are several methods of measuring this difference including Projected Difference Resolution (PDR)

The PDR metric is an analog to chromatographic resolution (12). PDR is applied to measure the separation of pairs of samples quantitatively in a multivariate data space. Given a GCMS data set that comprises two classes, the number of variables (i.e., number of data points, which is calculated by the number of retention time measurements times the number of mass-to-charge ratio measurements) is n, the numbers of objects (i.e., amount of GCMS spectra) in classes a and b are respectively m1 and m2. The data matrices Xa and Xb respectively have sizes of m1 × n and m2 × n, in which each row is a two-way GCMS data. The PDR measure of class separation Rs(a, b) is a scalar calculated by

$$\mathrm{R_s(a,b)} = \frac{|\bar{\mathbf{t}}_\mathbf{a} - \bar{\mathbf{t}}_\mathbf{b}|}{2(s_\mathrm{a} + s_\mathrm{b})} \qquad (1)$$

for which ta and tb are the scores for the two classes obtained by projecting the objects onto the difference vector $\overline{\mathbf{X}_\mathbf{a}} - \overline{\mathbf{X}_\mathbf{b}}$ of the class averages, given by

$$\mathbf{t}_\mathbf{a} = \mathbf{X}_\mathbf{a}(\overline{\mathbf{X}_\mathbf{a}} - \overline{\mathbf{X}_\mathbf{b}})^\mathrm{T} \qquad (2)$$

$$\mathbf{t_b} = \mathbf{X_b}(\overline{\mathbf{X_a}} - \overline{\mathbf{X_b}})^{\mathrm{T}} \quad (3)$$

for which $\overline{\mathbf{X_a}}$ and $\overline{\mathbf{X_b}}$ are the average class vectors that have a length of n. The column vectors ta and tb have lengths of m1 and m2, respectively. From the projections the averages $\overline{t}_a$ and $\overline{t}_b$ and their corresponding standard deviations s_a and s_b are calculated.

PDR is proposed as a straightforward multivariate measure for rapidly quantifying the separation of multivariate data objects for a pair of classes. The smaller the PDR, the harder to predict two classes by multivariate pattern recognition methods. Generally, a well-resolved separation of two classes has a PDR value greater than 1.5, which is comparable to the minimum resolution for baseline resolution between a pair of chromatographic peaks. When the data set contains more than two classes, the PDR metric for each pair of classes is systematically calculated for all combinations of pairs. The PDR matrix can be viewed as a triangle that measures the separation of each pair of classes.

Column bleeding was observed in the measurement of kerosene samples because of the higher column temperatures that are required for elution, which may bias the pattern classification of the two-way profiles. As a result, the baseline was corrected by a procedure based on the PCA result. For a two-way GCMS data matrix X, mass spectrometry scans are stored by rows and extracted ion chromatograms are stored by columns. First, the spectrum segment of the final one minute retention time window was selected to acquire background mass spectral scans. The PCA is performed on this background matrix of mass spectra for each sample. By performing the classification based on background subtraction using 1–10 largest principal components, it is concluded that the loading of the first principal component v1 characterize the mass spectrum of column bleeding impurities. Therefore, background correction is obtained by

$$\mathbf{X_c} = \mathbf{X} - \mathbf{X} \cdot \mathbf{v_1} \cdot \mathbf{v_1}' \quad (4)$$

for which Xc is the baseline corrected spectrum.

After baseline correction a third order polynomial was used to adjust the retention times of each chromatogram to the reference chromatogram using a two-way layout in the full retention time times the mass spectral image is used in the alignment process. The nonlinear simplex algorithm was used to maximize the correlation coefficient between the reference and each GCMS measurement. One sample (K10) was used as the reference chromatogram for retention time (RT) correction, however, several tests performed using other reference samples gave little difference in the final result. The RT correction was applied only to choose the correct target compounds in the AMDIS peak list report in the target compound ratio method. Because in the two-way profile method, peak binning reduces the effect of retention time drift, the RT correction did not improve peak identification.

Two approaches were applied to determine the number of latent variables in the partial least squares-discriminant analysis (PLS-DA) (12) method. In the optimal partial least squares-discriminant analysis (oPLS-DA), the number of latent variables is determined by achieving highest prediction accuracy for the prediction set. As a result, oPLS-DA is positively biased, which is applied as a reference method.

The other PLS-DA method is unbiased because the prediction set is not used to determine the number of latent variables in the PLS-DA model. The procedure for unbiased PLS-DA training is similar to the BLP method (13). First, the training set is split into two subsets using Latin partitions. Then, each partitioned subset is used once for prediction and once for model-building. The procedure is bootstrapped 10 times. Lastly, the prediction accuracies are averaged across the 10 bootstraps, the number of latent variables is determined by achieving highest average prediction accuracy.

FuRES is based on the decision tree algorithm and fuzzy logic theory, where each branch of the decision tree model is a multivariate fuzzy rule (14). FuRES has been successfully applied

in forensic researches to analyze two-way GCMS and gas chromatography–differential mobility spectrometry (GC–DMS) data (15,16).

OPLS-DA, PLS-DA, and FuRES were validated by the BLP method (18). The PDRs of each training set is calculated for comparison as well. Bootstrapping is a re-sampling method. Latin partition is a block cross-validation, in which the class distributions are maintained between the training set and the prediction set. BLP provides validation results with confidence intervals by running the evaluation repeatedly with different training and prediction set partitions. The MATLAB scripts for the kerosene analysis are given in Appendix 3.

For MDP data, peak area ratio data was analyzed using statistical routines (18) written in the open source statistical programming language R (R Development Core Team. "R: A Language and Environment for Statistical Computing". R Foundation for Statistical Computing (Vienna, Austria); http://www.R-project.org, 2012) (19). The R scripts utilized are given in Appendix 4.

RT drift was checked by running an E1618 standard mix of normal hydrocarbons with each set of samples. The retention times of the standard mix were compared with the normal hydrocarbons in representative samples from each set of samples Because the same column was used throughout the many months of data collection without removal/replacement, negligible changes in RT were observed. Where drift was observed, appropriate changes in the retention time windows were made. A QC sample was routinely analyzed to also check for drift.

Evaporation study

The QA/QC composite kerosene sample was evaporated to 25%, 50%, 75%, and 90% by volume under a stream of dry nitrogen gas at low heat on a hot plate.

Burn Tests

1.0 mL kerosene was dispensed on ~2x2" (5x5 cm) squares of wood, carpet, or carpet pad, ignited with a butane lighter, and allowed to self-extinguish. Kerosene residue was tested using the passive adsorption (ACS) ASTM Method E1412 with carbon disulfide as the extraction solvent.

Kimwipes® tissues spiked with 20µL of neat kerosene samples, K005 and K006, were tested using the ASTM Method E1412 for adsorption or desorption issues. Also, unburned and uncontaminated pieces of wood, carpet, and carpet pad were tested using the ASTM Method E1412 to see if overlapping data would occur in the kerosene elution region.

III. Results:

Kerosene Analysis

Forty-four kerosene samples were analyzed using target compound ratio analysis and 36 target compounds were identified, corresponding to 35 sequential ratios in each kerosene sample, as seen in Table 1.

Table 1. Target compound list, estimated retention time, and corresponding ratios identified in each kerosene sample. Specific isomers were not able to be identified in several cases due to lack of authentic standards. Qion was the ion used for peak area determination, given presence of the three qualifier ions and within the preset RT window.

					IONS (m/z)			
Peak #	Compound	Nominal RT	Ratio	Peak Ratio	Qion	Qual1	Qual2	Qual3
1)	Toluene	7.979	2/1	1	91	65	57	281
2)	p-Xylene	10.203	3/2	2	91	106	105	77
3)	Nonane	10.746	4/3	3	57	85	71	56
4)	1-ethyl-2-methyl-Benzene	12.575	5/4	4	105	120	91	77
5)	ethyl-methyl-benzene	12.683	6/5	5	57	105	71	120
6)	a-trimethyl-benzene	12.763	7/6	6	105	120	77	91
7)	ethyl-methyl-Benzene	13.18	8/7	7	105	120	91	77
8)	Decane	13.415	9/8	8	57	71	85	55
9)	1,2,4-trimethyl-Benzene	13.575	10/9	9	105	120	77	119
10)	A trimethyl-Benzene	14.495	11/10	10	105	120	77	91
11)	A diethyl-Benzene	15.255	12/11	11	57	119	105	71
12)	3-methyl-Decane	15.478	13/12	12	57	71	85	126
13)	Undecane	16.255	14/13	13	57	71	85	56
14)	A methyl-trans-Decalin	17.667	15/14	14	81	67	95	152
15)	Compound a	17.998	16/15	15	71	105	106	57
16)	2-methyl-Undecane	18.113	17/16	16	57	71	85	127
17)	Compound b	18.204	18/17	17	105	152	91	95
18)	methyl-ethyl-Benzene	18.358	19/18	18	119	134	57	85
19)	1,2,3,4-tetrahydro-Napthalene	18.816	20/19	19	104	132	91	115
20)	Dodecane	19.119	21/20	20	57	71	85	56
21)	Compound c	19.239	22/21	21	97	55	69	131
22)	2,6-dimethyl-Undecane	19.553	23/22	22	57	71	98	56
23)	Compound d	20.359	24/23	23	104	146	91	131
24)	Compound e	20.805	25/24	24	105	162	71	91
25)	A dihydro-dimethyl-1H-Indene	21.21	26/25	25	131	146	115	91
26)	1,2,3,4-tetrahydro-6-methyl-Napthalene	21.748	27/26	26	131	118	146	117
27)	Tridecane	21.896	28/27	27	57	71	85	55
28)	A tetrahydro-dimethyl-Napthalene	23.199	29/28	28	118	145	160	105
29)	Tetradecane	24.548	30/29	29	57	71	85	55
30)	1,7-dimethyl-Napthalene	26.086	31/30	30	156	141	57	71
31)	2,6,10,14-tetramethyl-hexadecane	26.2	32/31	31	57	71	85	141
32)	3-methyl-tetradecane	26.394	33/32	32	57	71	85	183
33)	Pentadecane	27.069	34/33	33	57	71	85	55
34)	Hexadecane	29.452	35/34	34	57	71	85	55
35)	Heptadecane	31.709	36/35	35	57	71	85	207
36)	Octadecane	33.847			57	71	207	85

Baseline correction

The TIC chromatograms of a kerosene sample are given in Figure 2. The effect of baseline correction is demonstrated. In kerosene samples, the baseline goes upwards in the uncorrected TIC profile. Compared to the uncorrected spectra, the baseline of the corrected spectra is improved. The pattern classification and PDR metric of the spectra before baseline correction is

performed. Comparisons were made between baseline corrected spectra and original spectra. The results are given in Table 2 below.

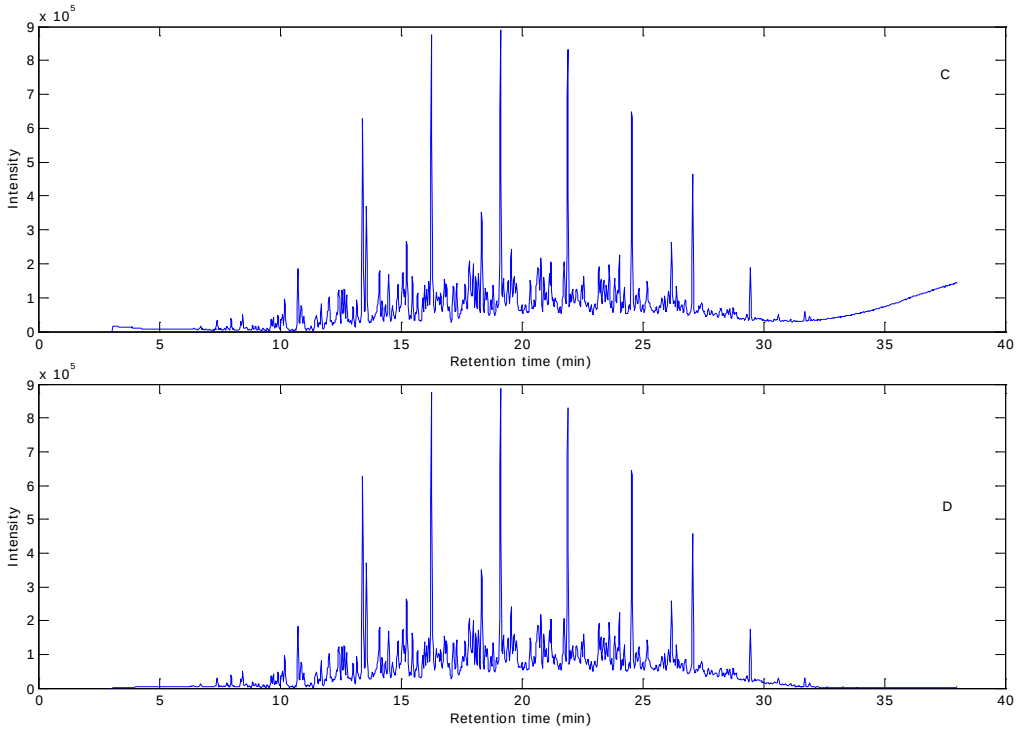

Figure 2. TIC chromatograms of a kerosene sample (C): before baseline correction, (D) after baseline correction.

PDR mapping

The PDR mapping of kerosene samples by the two-way profile method is given in Figure 3. The geometric mean of PDRs, plotted in grayscale, are measured repeatedly by removing one replicate from each class, a total of nine combinations of subsets for a pair of classes. The darkness of the box indicates the PDR value. All PDR values that are greater than or equal to 5 are plotted in white. The numbers printed in the box are the number of times out of a total of 60 times that an object was misclassified between the pair of classes during the BLP validation by FuRES. Most of the misclassifications of the classes are located in gray boxes, indicating that

the PDR metric effectively measures the predictive ability of the classifiers. It can be concluded the lower the PDR between two classes, the more likely misclassification will occur.

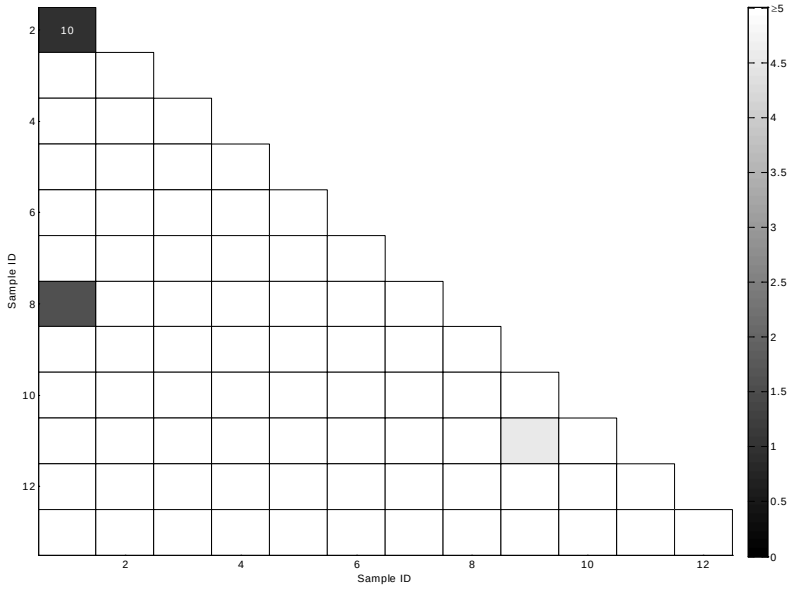

Figure 3. The PDR mapping of kerosene samples by the two-way profile method. The PDR values and the FuRES prediction use different bootstrap approaches. The PDR values are encoded by grey scale, which is the geometric mean of all possible subsets of Latin partitions. All PDR values that are greater than or equal to 5 are plotted in white. In a pair of classes that comprised of six objects, the subsets that comprised of four objects were obtained by removing one out of three objects in each class, which results nine possible combinations of subsets. The numbers in the box are the numbers of misclassifications between the corresponding pair of samples out of a total of 60 times by the BLP validation of the FuRES model.

Pattern classification

The BLP validation of PDR metric, oPLS-DA, PLS-DA, and FuRES are given in Table 2. The effect of baseline correction is evaluated. Although the prediction accuracy is not improved for two-way profiles after baseline correction, the PDRs were improved significantly in the kerosene spectra, indicating that the separation between each pair of classes was generally improved.

Table 2. PDRs and prediction accuracies of oPLS-DA, PLS-DA and FuRES with 95% confidence intervals by BLP validation. Both full two-way profile and component ratio methods are reported.

	Kerosene
Total number of objects	39
Two-way profile, original spectra	
Geometric mean PDR	17 ± 8
oPLS-DA (%)	100 ± 0
PLS-DA (%)	83 ± 6
FuRES (%)	97 ± 0
Two-way profile with baseline correction	
Geometric mean PDR	41 ± 15
oPLS-DA (%)	100 ± 0
PLS-DA (%)	92 ± 5
FuRES (%)	97 ± 0
Component ratio	
Geometric mean PDR	9 ± 2
oPLS-DA (%)	81 ± 7
PLS-DA (%)	62 ± 5
FuRES (%)	91 ± 6

Both the two-way profile and component ratio methods achieved prediction accuracies greater than 90% using the FuRES classifier. For the gasoline data set, the two-way profile method and the component ratio method performed equally well. The two-way profile method achieved higher prediction accuracies than the component ratio method for the kerosene data set because the two-way profiles retain more chemical information. The loss of peak information manifests itself in lower PDR values. The PDRs of the component ratio method is lower than the two-way profile method for both gasoline and kerosene data. It is essentially a differential transformation so there is a loss in signal-to-noise ratio, which can be expected with any differential transformation.

The two-way spectra contain noise. As a positively biased method, oPLS-DA achieved higher prediction accuracies for the data because of overfitting the data. FuRES is a soft classifier and is inherently resistant to overfitting. However, for the component ratio method overfitting is mostly avoided because the training data set is overdetermined (i.e., fewer variables

than objects). As a result, the FuRES method achieved better predictions for the component ratio data than the biased oPLS-DA method. The unbiased PLS-DA method achieved marginally better prediction accuracies for the two-way gasoline data that are statistically insignificant (15). Unbiased PLS-DA performs worse than the FuRES method for the rest of the data sets, especially in the classification of component ratio data. The performance demonstrated that FuRES is a powerful classifier for samples measured by GCMS.

Middle Petroleum Distillates

High resolution GCMS analyses for 155 MPD samples were collected in triplicate. Forty-one compounds were identified resulting in 33 peak area ratios. Not all compounds were found in all samples. Where a compound was not detected, its corresponding ratio was set to zero if it was in the denominator (normally would result in "divide by zero" error). Some ratios had %RSD exceeding the 5% nominal cut-off. The list of compounds and peak area ratios is given in Table 3.

Table 3. MDP target compounds, nominal retention times, peak ratios and monitored ions. Specific isomers were not able to be identified in several cases due to lack of authentic standards. Qion was the ion used for peak area determination, given presence of the three qualifier ions and within the preset RT window.

					Ions (m/z)			
Peak #	Compound	Nominal RT	Ratio	Peak Ratio	Qion	Qual1	Qual2	Qual3
1)	Octane	6.530	1	2/1	85	57	71	56
2)	1,3,5-trimethyl-cyclohexane.	7.251	2	3/2	111	69	55	126
3)	ethyl-cyclohexane	7.361	3	4/3	83	55	82	112
4)	2-methyl-octane	7.448	4	5/4	57	71	85	84
5)	3-methyl-octane	7.579	5	6/5	57	56	98	99
6)	1α,2β,4β-trimethyl- cyclohexane	7.645	6	7/6	111	69	55	126
7)	p-Xylene	7.711	7	17/7	91	106	105	77
8)	Nonane	7.995	8	10/8	57	85	71	56
9)	1,2,3-trimethyl- cyclohexane	8.082	9	14/9	69	111	55	126
10)	1-ethyl-4-methylcyclohexane.	8.191	10	11/10	97	55	126	69
11)	2,5-dimethyl-octane	8.497	11	12/11	57	71	85	70
12)	2,6-dimethyl-octane	8.672	12	13/12	57	71	105	56
13)	2,3-dimethyl-octane	8.913	13	14/13	57	98	55	56
14)	propyl-cyclohexane	9.000	14	17/14	83	55	82	126
15)	4-methyl-nonane	9.153	15	15/14	57	70	71	98
16)	3-methyl-nonane	9.372	16	16/15	57	105	71	56
17)	propyl-benzene	9.263	17	18/17	91	120	65	92
18)	1-ethyl-2-methyl-benzene	9.372	18	19/18	105	57	120	71
19)	1,2,3-trimethyl-benzene	9.503			105	120	119	77
20)	Decane	9.897	19	30/20	57	71	85	56
21)	1,2,4-trimethyl-benzene	10.115	20	23/21	105	97	120	55
22)	Unknown alkyl aromatic	10.421	21	22/20	71	57	70	55
23)	1,3,5-trimethyl-benzene	10.837			105	120	57	71
24)	butyl-cyclohexane	11.099	22	25/24	83	55	82	67
25)	5-methyl-decane	11.186	23	26/25	57	85	98	71
26)	2-methyl-decane	11.317	24	27/26	57	71	85	55
27)	3-methyl-decane	11.514			57	71	85	56
28)	1,2-diethyl-benzene	11.623			105	119	134	97
29)	1-methyl-2-propyl-benzene	11.776	25	30/29	105	134	0	0
30)	Undecane	12.126	26	37/30	57	71	85	56
31)	1,2,3,4-tetramethyl-benzene.	13.001	27	32/31	119	134	105	91
32)	1,2,3,5-tetramethyl- benzene	13.110			119	134	120	91
33)	4-methyl-undecane	13.613	28	34/33	71	57	70	85
34)	2-methyl-undecane	13.700	29	33/32	57	71	85	55
35)	3-methyl-undecane	13.919			57	71	85	56
36)	1,2,3,4-tetrahydro-napthalene.	14.465	30	37/36	104	132	91	57
37)	Dodecane	14.596	31	38/37	57	71	85	55
38)	2,6-dimethyl-undecane	14.968			57	71	98	56
39)	2-methyl-dodecane	16.214			57	71	85	99
40)	Tridecane	17.110	32	40/37	57	71	85	55
41)	Tetradecane	19.624	33	41/40	57	71	85	55

Chemometric Analysis of MPD target compound peak area ratios.

Three metrics of association between sets of MPD peak ratios (referred to as MPD chromatograms for brevity) were examined to test the efficacy of "matching" the chromatogram of an unknown MPD sample to known MPD. A (machine) association is the numerically based

comparison of an arbitrate pair of chromatograms. In this study a pair is deemed to "match" (or more appropriately stated, "be associated") by the algorithm if the value of the association metric exceeds a predefined cut-off. False positive and true positive rate estimates of the "matching-system" can be assessed at each cut-off using a large set of chromatograms with truly known identities. For this study a sample of 465 chromatograms was used. There were 155 unique MPDs within this sample. A total of 107,880 unique pairwise comparisons were then generated and separated into known-matching (KM) and known-non-matching (KNM) categories.

The metrics of association employed were the well-known Pearson correlation coefficient, Spearman's-rho rank correlation coefficient and Kendall's-tau coefficient (18). The performance of these functions was assessed by computing the true positive rate (i.e. the "hit-rate" or one-hundred minus the false negative rate) at what we consider a reasonable false positive rate of association in a practical forensic science application ~5%. While all three association metrics performed well, the worst was Pearson correlation coefficient (also commonly known as the Pearson product moment correlation coefficient). For a false positive rate of 5.0% ($R=0.53$) the hit-rate was 91.8%. The Spearman measure performed significantly better. For a false positive rate of 5.0% ($\rho=0.60$) the hit rate was 96.1%. Kendall's tau performed even a bit better. The receiver operating characteristic (ROC) curve appears below in Figure 4.

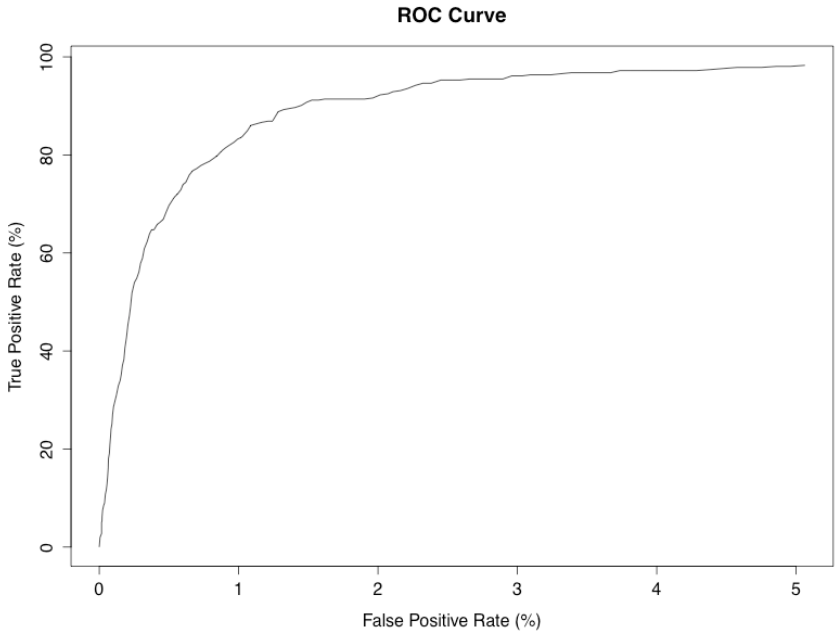

Figure 4. ROC curve for Kendall's-tau association metric for MPD peak ratio data.

For a false positive rate of 5.1% (τ=0.50) the hit rate was 98.3%. As can be seen from Figure 4, even at a 1% false positive rate, the hit-rate is still rather high at ~80%. Overall the average pairwise decision error rate for the best performing Kendall's-tau metric (i.e. on average how often can an incorrect conclusion of association/no-association be expected?) was 4.9%. Overall, this study indicates a sample of MPD can be associated with a known with high accuracy (~90%-95%), especially using the Kendall's-tau coefficient as an association metric. An alternative graphical representation is given in Figure 5. The KM are shown in red, the KNM are shown in green with the frequency of occurrence in the Y axis (density).

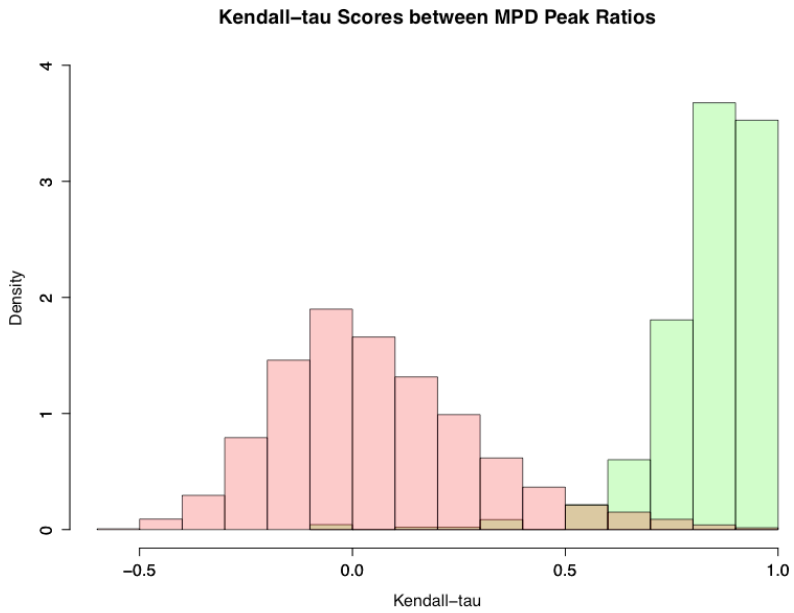

Figure 5. Kendall's-tau metric for "association" (red) vs. "no-association" (green) for MPD Peak ratio data.

Evaporation study:

As shown in Figure 6, evaporation of kerosene leads to a loss of early eluting compounds. Peak area ratios computed from these components will generally not be affected until the levels of recovered material is low, as shown in Figure 7.

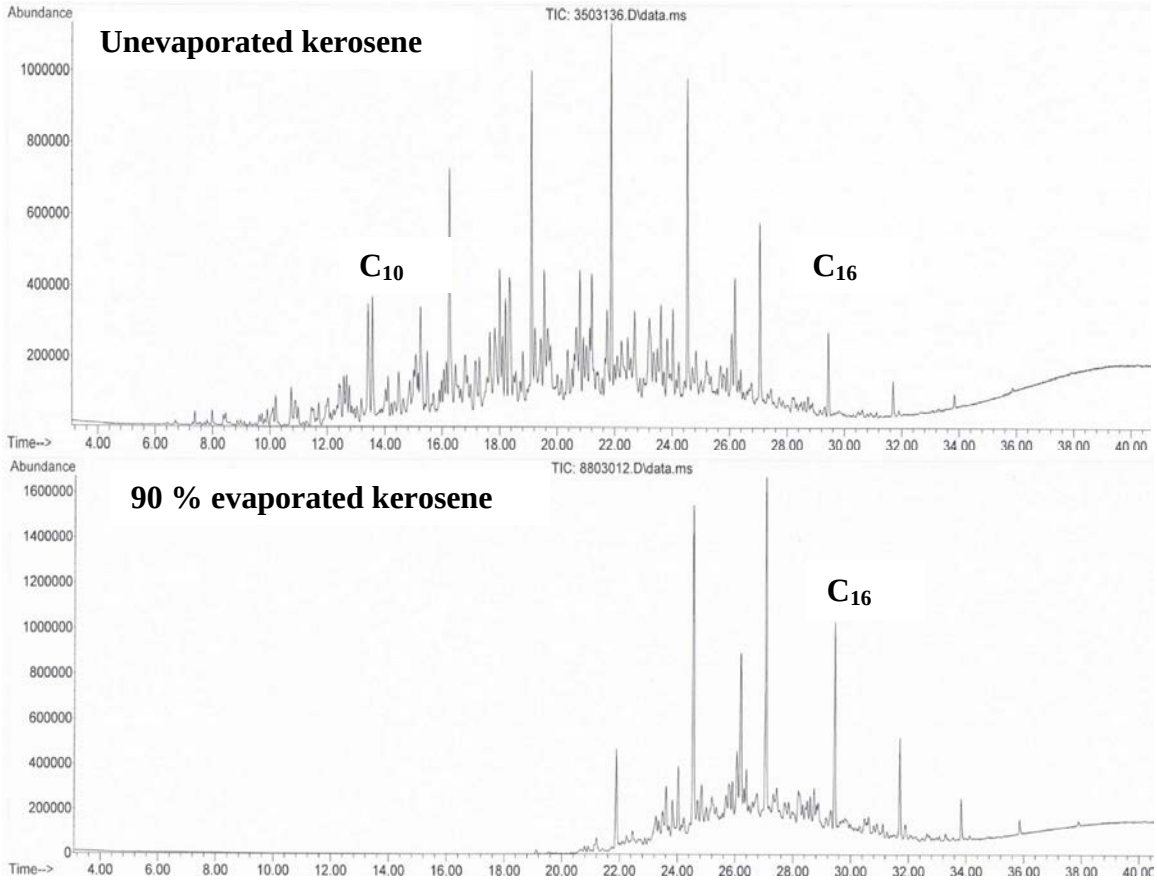

Figure 6. Comparison of unevaporated kerosene (upper) and 90 % evaporated kerosene.

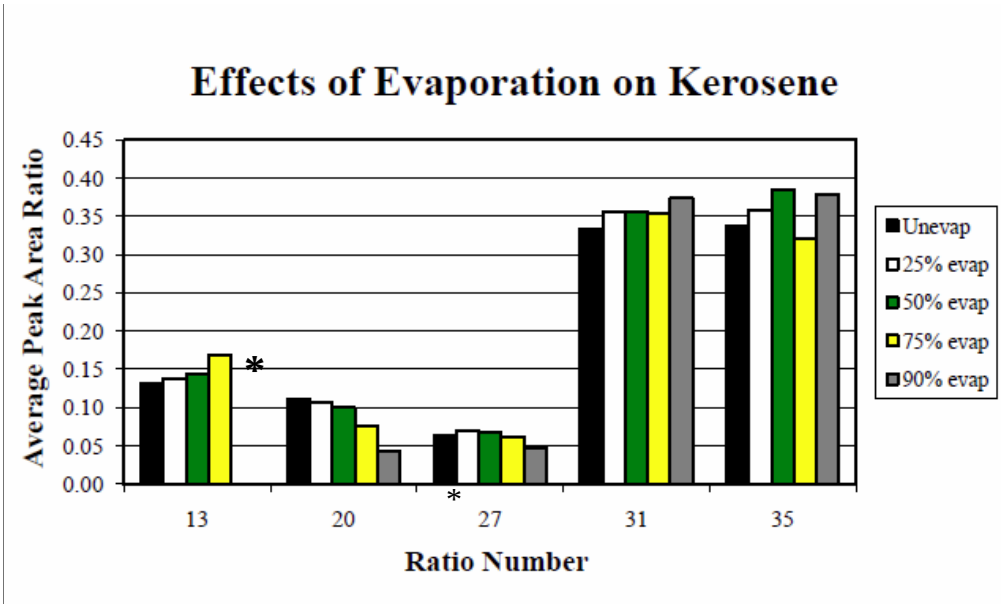

Figure 7. Five selected target compound ratios at different levels of evaporation. Asterisk indicates ratio had RSD >5% cutoff.

Burn Tests

Controlled burns of two kerosene samples were conducted on wood, carpet, and carpet pad. Target compound area ratios of neat samples on Kimwipe® (using the ASTM E1412 method) were similar to the neat injection samples, confirming no adsorption/desorption issues with the activated charcoal strip. Pyrolysis products were identified in all of the substrates, but the wood substrate was seen to have the least pyrolysis products. Target compound ratios were similar when comparing neat kerosene with diluted kerosene and spiked Kimwipe® demonstrating that E1412 method had little effect. However, when comparing spiked Kimwipe® with burn samples some differences were seen, especially with pine where normal paraffins were reduced relative to other components in the sample.

IV. Conclusions

*Discussion of findings***.**

In this study, different IL samples were identified using several chemometric techniques. PDR measured the separation between the different samples and the results were presented as heat maps. PLS-DA and FuRES was used to build classification models. The models were validated by BLP validation. FuRES for kerosene data sets predicted the classes with great than 90% accuracy. Furthermore, the results of PDRs and pattern classifications were consistent. The results indicated the usefulness of various chemometric methods including baseline correction by PCA, PCT, PDR, PLS-DA, and FuRES to the forensic analysis workflow of the IL identification task by GCMS. A novel method, PDR mapping, is presented for the first time for characterizing complex data sets.

The work also demonstrates the usefulness of both the two-way profile and component ratio methods. The PCT compressed two-way profile keeps both the gas chromatography and mass spectrometry information, which is useful in comparing unevaporated samples. Although less accurate in kerosene sample prediction, the component ratio method has been shown to an

effective method that provides an approach to compare unevaporated gasoline samples. Future work will involve the identification of ILR from fire debris by chemometrics, as well as to perform a feature selection study on the choice of component peaks.

For MPD, the Kendall-tau method did provide comparable results (~ 5% 'false positive rate') using target compound peak area ratios on unevaporated samples. Future work with evaporated samples and comparison to simulated fire debris is necessary before implementation in casework.

*Implications for policy and practice***.**

Application of high resolution GCMS analyses for comparison between neat ignitable liquids is possible using pattern matching techniques described in this report with some reasonable degree of statistical validity. With a reference database of high resolution GCMS analyses of gasoline, kerosene and MPD samples such as from the ILRC should allow individual laboratories to utilize these techniques without having to invest considerable time, manpower and expense to recreate the data. Key to adoption of such a database will be additional studies incorporating different instrumentation and laboratories utilizing a 'round robin' approach.

Implications for further research.

Initial studies in our laboratory have shown that there may be some effects due to substrate and/or evaporation, thus additional studies are needed to test the 'robustness' of this method before application to casework. Alternative chemometric methods should be investigated using this database of analyses in search of improved discriminatory ability. The database of target ratio analyses will enable other researchers to compare chemometric methods on the same dataset.

V. Acknowledgements

I wish to thank the personnel and management of Marathon-Ashland Refinery, Catlettsburg, Kentucky for generously providing kerosene samples and other refinery products used in this research. I also wish to thank Dr. Nicholas Petraco, John Jay College of Science and Criminal Justice, CUNY, New York City, NY for the statistical evaluation of the MPD data.

I also thank all the graduate students who contributed to this project and were supported by this grant: from Marshall University Forensic Science Program: Alexandra Bondra, MSFS; Carolyn Trader Moore, MSFS; Amanda Heeren, MSFS; Jared Vititoe, MSFS; Dana Greely, MSFS and Kendra Wilbur, MSFS. From the Department of Chemistry, University of Ohio, Athens: Weiying Lu, PhD and my co-PI and director of Weiying's research, Peter deB. Harrington, PhD, Professor.

V. References

1. Stauffer, E; Dolan, J.A; Newman, R., Fire Debris Analysis. Massachusetts: Elsevier, 2008.
2. Petraco NDK, Gil M, Pizzola PA, Kubic TA, Statistical Discrimination of Liquid Gasoline Samples from Casework. J Forensic Sci 2008; 53(5):1092-1101.
3. American Society for Testing Materials ASTM E1618-10, Standard test method for ignitable liquid residues in extracts from fire debris samples by gas chromatography-mass spectrometry. 2010.
4. Committee on Identifying the Needs of the Forensic Sciences Community, National Research Council, Strengthening forensic science in the United States: A path forward. Washington, DC, 2009.
5. DeHaan JD, Icove DJ, In Kirk's fire investigation, 7th ed.; Pearson: Upper Saddle River, NJ, 2011; p 679.

6. Mann DC, Comparison of Automotive Gasolines Using Capillary Gas Chromatography II: Limitations of Automotive Gasoline Comparisons in Casework. J. For. Sci. 1987 32(3): 6126-628.
7. Keto RO. GCMS Data Interpretation for Petroleum Distillate Identification in Contaminated Arson Debris. J Forensic Sci 1995; 40(3):412-423.
8. Dolan JA, Ritacco CJ. Gasoline comparisons by gas chromatography/mass spectrometry utilizing an automated approach to data analysis. Proceedings of the Annual Meeting of the American Academy of Forensic Sciences; 2002 Feb 11–16; Atlanta, GA.
9. Barnes, AT, Dolan JA, Kuk RJ, Siegel JA, Comparison of Gasolines using Gas Chromatography-Mass Spectrometry and Target Ion Response. J Forensic Sci 2004; 49(5):1-6.
10. Allen SP, Williams MR, Bryant C, Byron D, Cerven J, Cooper BD, Hilliard DC, Hoffmann J, Kwast J, Thomas SA, Whitcomb CM, The National Center for Forensic Science Ignitable Liquids Reference Collection and Database, Forensic Science Comm. 2006; 8(2).
11. Brereton, RG. Chemometrics: Data Analysis for the Laboratory and Chemical Plant. Wiley: Chichister, W. Sussex, England, 2003.
12. Lu Y, Chen P, Harrington PB, Comparison of differential mobility spectrometry and mass spectrometry for gas chromatographic detection of ignitable liquids from fire debris using projected difference resolution, *Anal. Bioanal. Chem.* **2009;,** *394(8)*: 2061-2067.
13. Harrington PB, Kister J, Artaud J, Dupuy J, Automated principal component-based orthogonal signal correction applied to fused near infrared-mid-infrared spectra of French olive oils, *Anal. Chem.* **2009,** *81(17)*, 7160-7169.
14. Harrington PB, Fuzzy multivariate rule-building expert systems - minimal neural networks, *J. Chemometr.* **1991,** *5(5)*, 467-486

15. Rankin JG, Bondra A, Trader C, Lu W, Harrington P. Target compound ratios and chemometric analyses for the individualization of neat ignitable liquids and residues from fire debris, *Proceedings of the 12th International Interflam Conference*, London, July 5-7, 2010; Interscience Communications: London, **2010**; pp 1305-1320.
16. Lu Y, Chen P, Harrington PB,, Forensic application of gas chromatography - differential mobility spectrometry with two-way classification of ignitable liquids from fire debris, *Anal. Chem.* **2007,** *79(17)*, 6752-6759.
17. Harrington PB, Statistical validation of classification and calibration models using bootstrapped Latin partitions, *Trends Anal. Chem.* **2006,** *25(11)*, 1112-1124.
18. Sokal RR, Rohlf BJ, "Biometry". W.H Freeman: New York, 2012.
19. Curan, JM. Introduction to Data Analysis with R for Forensic Scientists. CRC Press, Boca Raton, 2011.

VI. Dissemination of Research Findings

Published Papers and Proceedings

Rankin JG, Bondra A, Trader C, Lu W, Harrington P. Target compound ratios and chemometric analyses for the individualization of neat ignitable liquids and residues from fire debris, *Proceedings of the 12th International Interflam Conference*, London, July 5-7, 2010; Interscience Communications: London, **2010**; pp 1305-1320.

Lu W, Rankin JG, Bondra A, Trader C, Heeren A, Harrington PB. Ignitable liquid identification using gas chromatography/mass spectrometry data by projected difference resolution mapping and fuzzy rule-building expert system classification. *Forensic Sci. Intern*. 2012: 218-222 http://dx.doi.org/10.1016/j.forsciint.**2012**.03.003

Dissertation

Wieying Lu, "Development of Radial Basis Function Cascade Correlation Networks and Applications of Chemometric Techniques for Hyphenated Chromatography–Mass Spectrometry Analysis", PhD Dissertation, Department of Chemistry, University of Ohio, Athens, November 2011. Pp153.

Presentations at National and Regional Meetings

Wilbur K, Rankin JG. "Target Compound Ratio Analysis of Medium Petroleum Distillates." AAFS 2012, Atlanta, GA, Feb **2012**, Poster Presentation, A116.

Rankin JG "Individualization of Ignitable Liquid Residues in Fire Debris", Tripartite Symposium, Pittsburgh, PA, May **2010**, Invited Paper.

Bondra A, Trader C, Rankin JG. "The Differentiation of Kerosene Samples by Target Compound Ratio Analysis", AAFS 2010, Seattle, WA, Feb **2010**, Poster Presentation, A103.

Database

A CD-ROM with the complete data sets for kerosene and MPD peak ratios in Excel format, R scripts used for MPD and MATLAB scripts for Kerosene data analyses will be submitted to NIJ in August 2012 for distribution to interested parties.

Appendix 1

Kerosene Sample List

Sample ID	Sequence #	Station	Location	Date	Designation	Description
K001	1563192	Marathon	Bay City MI	9/3/2008	ULK1	C-composite
K002	1563822	Marathon	Columbus OH Term W	9/4/2008	ULK1	C-composite
K003	1569763	Marathon	Muncie IN	9/15/2008	ULK1	C-composite
K004	1563185	Marathon	Brecksville Term	9/3/2008	ULK1	C-composite
K005	1565967	Marathon	Green Bay WI	9/8/2008	ULK1	C-composite
K006	1563776	Marathon	Huntington WV	9/4/2008	ULK1	B-Bottom
K007		Marathon	St. Paul Park MN	8/29/2008	ULSK	
K008	1568293	Marathon	Kinder Morgan Argo	9/2/2008	Kero-K1	
K009		BP	Huntington, WV	9/13/2004	Kero-K1	
K010	1580598	Marathon	Lexington KY	10/6/2008	ULK1	C-composite
K011	1578453	Marathon	Bay City MI	10/1/2008	ULK1	C-composite
K012	1578738	Marathon	Evansville IN	10/2/2008	ULK1	C-composite
K013	1577733	Marathon	Griffith IN	9/30/2008	ULK1	C-composite
K014	1578273	Marathon	Champaign IL	10/1/2008	ULK1	C-composite
K015		Marathon	St. Paul Park, MN	03/26/08	ULK1 Jet A	702
K016		Marathon	Lexington, KY	04/01/08	ULK1	703
K017		Marathon	Bay City MI	04/03/08	ULK1	704
K018		Marathon	Columbus, OH	03/20/08	JET-A	705
K019	1634560	Marathon	Louisville KY	1/20/2009	ULK1	B-Bottom
K020	1632354	Marathon	Jackson MI	1/14/2009	ULK1	C-composite
K021	1635730	Marathon	Kinder Morgan Argo	1/14/2009	ULK1	
K022	1630489	Marathon	Green Bay WI	1/12/2009	ULK1	C-composite
K023	1577173	Marathon	Louisville TN	9/22/2008	JET-A	
K024	1577174	Marathon	Louisville TN	9/22/2008	JET-A	
K025	1575177	Marathon	Columbus OH Term E	9/28/2008	JET-A	V2-#2 Clay Fitr out
K026	1571504	Marathon	Speedway	9/18/2008	JET-A	
K027	1575176	Marathon	Columbus OH Term E	9/28/2008	JET-A	V2-#2 Clay Fitr out
K028	1571506	Marathon	Speedway	9/18/2008	JET-A	
K029	1474987	Marathon	Columbus OH Term E	9/20/2008	JET-A	
K 030	1676347	Marathon	Milwaukee WI	4/13/2009	Kero-K1	C-composite
K 031	1677978	Marathon	Kinder Morgan Argo	4/14/2009	ULK1	
K 032	1676338	Marathon	Jackson MI	4/8/2009	ULK1	C-composite
K033	1629399	Marathon	Belton Term	1/29/2009	Kero-K1	Tk 351
K034			Kerosene Mix	6/29/2009	Not Evap	
K035			Kerosene Mix	6/29/2009	25% Evap	
K036			Kerosene Mix	6/29/2009	50% Evap	
K037			Kerosene Mix	7/23/2009	75% Evap	
K038			Kerosene Mix	7/29/2009	90% Evap	
K039	3023146	Marathon	CAN-Tanks Canton OH	8/13/2009	Kero-K1	
K040	3017670	Marathon	Champaign IL	8/7/2009	ULK1	C-composite
K041	3019941	Marathon	Lebanon OH	8/11/2009	ULK1	B-Bottom
K042	3023342	Marathon	Kinder Morgan Argo	8/10/2009	ULK1	
K043	3020109	Marathon	Green Bay WI	8/11/2009	ULK1	C-composite

K044		Marathon	MPC Clarksville IN	10/13/2009	ULK1		
K045	306993	Marathon	Lexington KY	10/14/2009	ULK1		C-composite
K046	3126411	Marathon	Knoxville Term-M	2/10/2010	Kero-K1	TK304	C-composite
K047	3128973	Marathon	Muncie	2/15/2010	ULK1	TK105	C-composite
K048	3127138	Marathon	Lebanon Term	2/11/2010	Kero-K1	TK2009	B-Bottom
K049	3127638	Marathon	Kinder Morgan Argo	2/8/2010	ULK1	TK 2517	
K050	3125866	Marathon	Jackson Term	2/8/2010	ULK1	TK153	C-composite
K051	3126334	Marathon	Lexington Term	2/10/2010	ULK1	TK828	C-composite
K052	3125378	Marathon	Green Bay Term	2/8/2010	ULK1	TK242	C-composite
K053	3122261	Marathon	Bay City	2/3/2010	ULK1	TK2	C-composite
K054	608-090	Lowes	Crown-Lowes	2/10/2009	Kerosene 1-K	IL 22	
K055	903-102	Lowes	Crown-Lowes OH	7/14/2009	Kero 1-k	IL 41	
K056	3289124	Marathon	Flint Term	11/9/2010	Kero-K1	TK202	C-composite
K057	3289828	Marathon	SSA Retailers	11/8/2010	Kero-K1		
K058	3272387	Marathon	Store 9633	11/7/2010	Kero-K1		
K059	3291104	Marathon	Store 9259	11/11/2010	Kero-K1		
K060	3286862	Marathon	Lebanon	11/4/2010	ULK1	TK2009	B-Bottom
K061	3271539	Marathon	Flar, KY	10/6/2010	Kero-K1		
K062	3288543	Marathon	Cincinnati, OH	11/8/2010	Kero-K1		
K063	3291112	Marathon	Store 9603	11/11/2010	Kero-K1		
K064	3287765	Marathon	Cincinnati Term	11/7/2010	ULK1	TK153	C-composite
K065	3271550	Marathon	Coving, KY	10/6/2010	Kero-K1		
K066	3306604	Marathon	Huntington WV	12/6/2010	Kero-K1		N-Nozzle
K067	3312867	Marathon	SSA Retailers	12/10/2010	Kero-K1		N-Nozzle
K068	3322431	Marathon	Cincinnati, OH	12/17/2010	Kero-K1		N-Nozzle
K069	3312779	Marathon	Ironton, OH	12/16/2010	Kero-K1		N-Nozzle
K070	1	Marathon	St Paul Park, MN	12/29/2010	ULSK	T-140	
K071	3310703	Marathon	Pora, WV	12/10/2010	Kero-K1		N-Nozzle
K072	3292947	Marathon	South Point, OH	11/12/2010	Kero-K1		N-Nozzle
K073	3291175	Marathon	Cincinnati, OH	11/9/2010	Kero-K1		N-Nozzle
K074	3291165	Marathon	Cincinnati, OH	11/9/2010	Kero-K1		N-Nozzle
K075	3305672	Marathon	Coving, KY	12/3/2010	Kero-K1		N-Nozzle
K076	3312781	Marathon	South Point, OH	12/16/2010	Kero-K1		N-Nozzle
k077	3326174	Marathon	Brecksville Term	1/11/2011	ULK1	TK359	R-Rack
k078	3326174	Marathon	Brecksville Term	1/11/2011	ULK1	TK359	R-Rack
k079	3326173	Marathon	Brecksville Term	1/11/2011	ULK1	TK359	C-composite
k080	3326173	Marathon	Brecksville Term	1/11/2011	ULK1	TK359	C-composite
k081	3326173	Marathon	Brecksville Term	1/11/2011	ULK1	TK359	C-composite
k082	3326174	Marathon	Brecksville Term	1/11/2011	ULK1	TK359	R-Rack
k083	3414992	Marathon	Columbus, OH Term	6/10/2011	Kero-K1	TK374	R-Rack
k084	3339438	Marathon	Huntington WV	2/3/2011	Kero-K1		N-Nozzle
k085	3364902	Marathon	Sharonville, OH	3/15/2011	Kero-K1		N-Nozzle
k086	3346454	Marathon	Lawrenceburg, IN	2/14/2011	Kero-K1		N-Nozzle
k087	3356035	Marathon	Huntington WV	3/2/2011	Kero-K1		N-Nozzle
k088	3366499	Marathon	Marietta, OH	3/2/2011	Kero-K1		N-Nozzle
k089	3364901	Marathon	Latonia, KY	3/16/2011	Kero-K1		N-Nozzle
k090	3344226	Marathon	Louisa, KY	2/3/2011	Kero-K1		N-Nozzle
k091	3348578	Marathon	Evendale, OH	2/15/2011	Kero-K1		N-Nozzle

k092	3369111	Marathon	South Point, OH	3/21/2011	Kero-K1	N-Nozzle
k093	3347383	Marathon	Ironton, OH	2/16/2011	Kero-K1	N-Nozzle
k094	3356002	Marathon	Ashland, KY	3/2/2011	Kero-K1	N-Nozzle
k095	3371850	Marathon	Charleston, WV	3/28/2011	Kero-K1	N-Nozzle
k097	3347161	Marathon	Latonia, KY	2/14/2011	Kero-K1	N-Nozzle
k098	3353762	Marathon	Charleston, WV	2/24/2011	Kero-K1	N-Nozzle
k099	3371856	Marathon	Poca, WV	3/28/2011	Kero-K1	N-Nozzle
k100	3405063	Marathon	Peotone, IL	5/16/2011	Kero-K1	N-Nozzle
k101	3364854	Marathon	Evendale, OH	3/15/2011	Kero-K1	N-Nozzle
k102	3353775	Marathon	Poca, WV	2/24/2011	Kero-K1	N-Nozzle
k103	3360034	Marathon	Cinti, OH	2/9/2011	Kero-K1	N-Nozzle

Appendix 2

Middle Petroleum Distillate Samples

MPD #	ID #	Brand Name	Sample Name
1	ILRC 003	Shellsol	Shellsol D38
2	ILRC 004	Shellsol	Shellsol D43
3	ILRC 011	SCCC	SCCC Mineral Spirits 145 EC
4	ILRC 013	Shellsol	Shellsol D60
5	ILRC 021	Parks	Parks 100% Mineral Spirit paint thinner
6	ILRC 024	Kingsford	Odorless charcoal lighter
7	ILRC 026	Klean Strip	100% Mineral Spirit paint thinner
8	ILRC 030	Klean Strip	Odorless mineral spirits
9	ILRC 031	Publix	Charcoal lighter
10	ILRC 034	Royal Oak	premium odorless charcoal lighter
11	ILRC 043	Chevron	Techron concetrate
12	ILRC 044	Pro-Gard	clean up
13	ILRC 045	Pro-Gard	fuel injector cleaner
14	ILRC 046	Pro-Gard	fuel injector plus intake valve cleaner
15	ILRC 047	Pro-Gard	gas treatment
16	ILRC 063	Whitaker	low end point mineral spirits
17	ILRC 064	Whitaker	paint thinner - mineral spirits
18	ILRC 065	Whitaker	Rule 66 mineral spirits
19	ILRC 081	Exxon	Varsol 1
20	ILRC 083	Exxsol	D 40 solvent
21	ILRC 084	Exxsol	D 60 Solvent
22	ILRC 085	Exxsol	D 80
23	ILRC 091	E-Z	paint thinner
24	ILRC 094	Flood	Penetrol (quality paint conditioner)
25	ILRC 095	Flood	ESP (Easy Surface Prep)
26	ILRC 101	Super G	charcoal lighter fluid - low odor
27	ILRC 123	Bruce	Clean n' strip
28	ILRC 129	Ace	premium quality charcoal lighter
29	ILRC 130	Ace	pure odorless mineral spirits
30	ILRC 134	WD-40	WD-40
31	ILRC 136	Cutter	citronella torch fuel
32	ILRC 160	Sunnyside	mineral spirits
33	ILRC 161	Sunnyside	mineral spirits, 25% weathered
34	ILRC 162	Sunnyside	mineral spirits, 50% weathered
35	ILRC 163	Sunnyside	mineral spirits, 75% weathered
36	ILRC 164	BBQ Pro	charcoal lighter fluid
37	ILRC 165	BBQ Pro	charcoal lighter fluid, 25% weathered
38	ILRC 166	BBQ Pro	charcoal lighter fluid, 50% weathered
39	ILRC 167	BBQ Pro	charcoal lighter fluid, 75% weathered
40	ILRC 178	ArmorAll	waterproofing sealer
41	ILRC 180	Outers	tri-lube

42	ILRC 181	Remington	Rem oil
43	ILRC 213	3M	08892 rust figher
44	ILRC 215	Auto Kare	SG-7 tire dressing
45	ILRC 221	Thompson's	waterseal
46	ILRC 222	Klean Strip	100% Mineral Spirit paint thinner
47	ILRC 223	Klean Strip	Odorless mineral spirits
48	ILRC 225	Ace	pure odorless mineral spirits
49	ILRC 227	Ace	paint thinner (100% mineral spirits)
50	ILRC 230	E-Z	low odor mineral spirits
51	ILRC 232	Kingsford	odorless charcoal lighter (can)
52	ILRC 233	Kingsford	odorless charcoal lighter (bottle)
53	ILRC 234	BBQ Pro	charcoal lighter fluid
54	ILRC 253	Formby's	build-up remover
55	ILRC 265	Klean Strip	Japan drier
56	ILRC 267	Kiwi Camp	dry heavy duty water repellent
57	ILRC 268	Jasco	adhesive clean-up
58	ILRC 275	Pro	paint thinner 100% mineral spirits
59	ILRC 276	USA	Lacquer thinner
60	ILRC 283	Kingsford	odorless charcoal lighter
61	ILRC 293	ProGard	fuel injector cleaner
62	ILRC 296	Tiki	torch fuel
63	ILRC 309	Piggly Wiggly	odorless charcoal lighter fluid
64	ILRC 312	Walmart	charcoal starter
65	ILRC 313	Sunnyside	mineral spirits
66	ILRC 315	Do It Best	Odorless mineral spirits
67	ILRC 316	Laura Lynn	charcoal starter
68	ILRC 325	Lamplight Farms	citronella torch fuel
69	ILRC 332	ArmorAll	waterproofing sealer
70	ILRC 339	Specs	paint thinner (low odor mineral spirits)
71	ILRC 340	HomeBest	odorless charcoal lighter fluid
72	ILRC 346	Ace	premium quality charcoal lighter
73	ILRC 349	Ace	pure odorless mineral spirits
74	ILRC 353	Klean Strip	Japan drier
75	ILRC 354	Longs	gourmet grill charcoal lighter
76	ILRC 355	Jasco	adhesive clean-up
77	ILRC 358	Mastercraft	varsol
78	ILRC 360	Royal Oak	charcoal lighter
79	ILRC 368	Mr. BarbQ	charcoal lighter
80	ILRC 371	Klean Strip	Odorless mineral spirits
81	ILRC 373	Ace	paint thinner 100% mineral spirits
82	ILRC 387	Rutland	tank shield fuel oil additive
83	ILRC 388	HomeBest	odorless charcoal lighter fluid
84	ILRC 389	Klean Strip	Klean heat odorless heater fuel
85	ILRC 393	Mineral Spirits	mineral spirits
86	ILRC 403	Exxon	charcoal lighter 105
87	ILRC 421	Turtle Wax	bug & tar remover

88	ILRC 425	TruBurn Kmart	odorless charcoal lighter
89	ILRC 431	Wizard	odorless double filtered charcoal starter
90	ILRC 432	Rainbow	odorless charcoal lighter fluid
91	ILRC 435	Ace	paint thinner 100% mineral spirits
92	ILRC 437	Kingsford	odorless charcoal lighter fluid
93	ILRC 438	Klean Strip	Odorless mineral spirits
94	ILRC 440	Ace	odorless charcoal lighter
95	ILRC 446	Stoddard	solvent
96	ILRC 452	Kingsford	odorless charcoal lighter
97	ILRC 465	Ace	paint thinner, 100% mineral spirits
98	ILRC 472	Pro-mix	2-cycle engine oil
99	ILRC 477	Minwax	wood finish
100	ILRC 478	Speedy Spar	varnish
101	ILRC 479	Majic	polyurethane enamel
102	ILRC 481	Homestar	citronella torch fuel
103	ILRC 488	Safeway	charcoal lighter
104	ILRC 495	Mastercraft	odorless paint thinner & all-purpose cleaner
105	ILRC 496	Mastercraft	low odor varsol
106	ILRC 499	Home	paint thinner
107	ILRC 500	Recochem	clear kerosene
108	ILRC 509	Prestone	cold start
109	ILRC 512	Watco	satin wax
110	ILRC 538	Kingsford	charcoal lighter
111	ILRC 541	Klean Strip	VM&P Naphtha
112	IL 5	Flood Paint Solutions	Penetrol
113	IL 6	Marathon	TK 163 MS (mineral spirits)
114	IL 10	Kingsford	Odorless Charcoal Lighter *
115	IL 12	Crown	low odor mineral spirits
117	IL 14	Crown	paint thinner *
119	IL 17	Crown	paint thinner *
122	IL 20	Daler-Rowney	low odor thinner
123	IL 24	Klean Strip	paint thinner
124	IL 25	Kingsford	odorless charcoal lighter *
125	IL 26	Kingsford	odorless charcoal lighter *
126	IL 28	E-Z	Paint thinner *
127	IL 29	E-Z	Paint thinner *
128	IL 30	Klean Strip	Paint thinner *
129	IL 32	Klean Strip	mineral spirits *
130	IL 33	Klean Strip	paint thinner *
133	IL 36	Klean Strip	Odorless mineral spirits *
134	IL 38	Klean Strip	paint thinner *
135	IL 39	Klean Strip	natural multi-purpose solvent
136	IL 40	Kingsford	odorless charcoal lighter *
137	IL 42	Crown	paint thinner - 100% mineral spirits
138	IL 43	Crown	low odor paint thinner
139	IL 44	Crown	low odor mineral spirits

140	IL 45	Kingsford	charcoal lighter *
141	IL 48	Kingsford	odorless charcoal lighter *
142	IL 49	Klean Strip	VM&P Naphtha
143	IL 50	Klean Strip	Safer paint thinner
144	IL 51	Klean Strip	Odorless mineral spirits *
145	IL 52	Klean Strip	paint thinner *
146	IL 53	Chefmat	charcoal lighter fluid
147	IL 54	Crown	paint thinner *
148	IL 55	Crown	low odor mineral spirits *
151	IL 58	Crown	liquid deglosser
154	IL 61	Klean Strip	Odorless mineral spirits *
158	IL 65	Crown	low odor mineral spirits *
159	IL 66	Kingsford	charcoal lighter fluid *
160	IL 67	Klean Strip	Odorless mineral spirits *
162	IL 69	Klean Strip	paint thinner w/ mineral spirits *

Notes: ILRC numbers refer to the Ignitable Liquid Reference Collection. Descriptions of these samples are taken from the ILRC database. "IL" numbers are samples purchased locally from home improvement stores. * denotes samples with similar names, but different lot numbers.

Appendix 3

MATLAB scripts utilized for the kerosene statistical processing

Main routine

```
RUN_ALL_KEROSENE
task=[1,0,1,1];
tic_is=0; %process Full two-way data (0)/Total ion chromatogram (1)/or Integrated spectra (2)
 %% select object, build pickflag for y-matrix
load ylabel_all_kerosene
samplei=[2]; %select batch to analyze
pickflag=(sum(yrundate(:,samplei),2)==1);
lookuptable=['K10';'K12';'K13';'K14';'K15';'K17';'K18';'K19';'K20';'K21';'K22';'K24';'K30'];
dirname='F:\Weiying Lu\Gasoline\Kerosene MS File\txtout';
curdir=cd;
cd(dirname);
filelist=dir('*.txt.mat');
cd(curdir)
m = length(filelist);
for i=1:m
   fname=filelist(i).name;
   pickflag(i)=0;
   for j=1:size(lookuptable,1)
      if  strcmp(fname(1:3),lookuptable(j,:))==1
         pickflag(i)=1;
      end
   end
end
j=1;
for i=1:m
      if pickflag(i)>0
      fname=filelist(i).name;
      idlist{j}=fname(1:3);
      j=j+1;
   end
end
idlist=unique(idlist)';
y=ysample(pickflag>0,:);
y(:,sum(y)==0)=[];
%% other parameter settings
global ms1;
% use retention time deminsion binning
minrt=3; maxrt=38; rtinc = 0.02;
grt = [minrt:rtinc:maxrt];
mzinc = 1;
mingmz=40; maxgmz=300;mzinc = 1;
gmz = [mingmz:mzinc:maxgmz];
cd(dirname);
```

```
load ([filelist(1).name]);
cd(curdir)
ind=(grt>38);
grt(ind)=[];

n = length(grt);
o = length(gmz);
rtlev=2; mzlev=2;
x=zeros(sum(pickflag),n*o);
xcnt=1;
%dirname='F:\Weiying Lu\Gasoline\Kerosene MS File\txtout';
dirname='C:\Users\Weiying Lu\Documents\Backup of my files on 64bit
computer\Gasoline\Kerosene MS File\txtout';
curdir=cd;
cd(dirname);
load('K34A.txt.mat');
cd(curdir)
        %% peak binning on mz dimension
        gcms1=zeros(size(gcms,1),length(gmz));
        maxgmz=max(gmz);
        mingmz=min(gmz);

        for mzi=1:length(mz)
           mzn=mz(mzi);
           if (mzn<=maxgmz && mzn>=mingmz)
              mzind=round((mzn-mingmz)./mzinc)+1;
              gcms1(:,mzind)=gcms1(:,mzind)+gcms(:,mzi);
           end
        end
        %% cut the rt range larger than 38 min
        ind=(rt>38);
        gcms1(ind,:)=[];
        %% peak binning on retention time deminsion
        gcms2=zeros(length(grt),length(gmz));
        maxgrt=max(grt);
        mingrt=min(grt);

        prev_rtind=1;
        cnt=1;
        for rti=1:length(rt)
           rtn=rt(rti);
           if (rtn<=maxgrt && rtn>=mingrt)
              rtind=round((rtn-mingrt)./rtinc)+1;
              gcms2(rtind,:)=gcms2(rtind,:)+gcms1(rti,:);

              if rtind==prev_rtind
                 cnt=cnt+1;
              else
                 gcms2(prev_rtind,:)=gcms2(prev_rtind,:)./cnt;
                 cnt=1;
```

```
            end
            prev_rtind=rtind;
        end
    end
    %correct last scan's bug
    gcms2(prev_rtind,:)=gcms2(prev_rtind,:)./cnt;
    %% baseline correction and column bleed correction using smartbaseline
    %find how many spectra in the 1 and 0.1 min interval
    [temp ind1]=min(abs((grt-grt(1))-1));
    [temp ind01]=min(abs((grt-grt(1))-0.1));
    %Use last num spectra for background correction, here use the first 1min and last 0.1 min
spectra
    backnum = [1:ind1, length(grt)-ind01:length(grt)];
    basisnum = 4;
    err = -inf;

    gcms3 = smartbaseline2(gcms2, backnum, basisnum, err);
    gcms3(gcms3<0)=0;

    refgcms=gcms3./norm(gcms3);

%% preprocess GC/MS spectra
if task(1)==1
  for i=1:m
    i
    if pickflag(i)==1
      %% load binned spectra
      cd(dirname);
      load ([filelist(i).name]);
      cd(curdir)

      %% peak binning on mz dimension
      gcms1=zeros(size(gcms,1),length(gmz));
      maxgmz=max(gmz);
      mingmz=min(gmz);

      for mzi=1:length(mz)
        mzn=mz(mzi);
        if (mzn<=maxgmz && mzn>=mingmz)
          mzind=round((mzn-mingmz)./mzinc)+1;
          gcms1(:,mzind)=gcms1(:,mzind)+gcms(:,mzi);
        end
      end
      %% cut the rt range larger than 38 min
      ind=(rt>38);
      gcms1(ind,:)=[];
      %% peak binning on retention time deminsion
      gcms2=zeros(length(grt),length(gmz));
      maxgrt=max(grt);
      mingrt=min(grt);
```

```
        prev_rtind=1;
        cnt=1;
        for rti=1:length(rt)
           rtn=rt(rti);
           if (rtn<=maxgrt && rtn>=mingrt)
              rtind=round((rtn-mingrt)./rtinc)+1;
              gcms2(rtind,:)=gcms2(rtind,:)+gcms1(rti,:);

              if rtind==prev_rtind
                 cnt=cnt+1;
              else
                 gcms2(prev_rtind,:)=gcms2(prev_rtind,:)./cnt;
                 cnt=1;
              end
              prev_rtind=rtind;
           end
        end
        %correct last scan's bug
        gcms2(prev_rtind,:)=gcms2(prev_rtind,:)./cnt;
        %% baseline correction and column bleed correction using smartbaseline
        [temp ind1]=min(abs((grt-grt(1))-1));
        %Use last num spectra for background correction, here use the last 1min spectra
        backnum = [length(grt)-ind1:length(grt)];
        basisnum = 1;
        err = -inf;

        gcms3 = smartbaseline2(gcms2, backnum, basisnum, err);
        gcms3(gcms3<0)=0;

        %% save result
        x(xcnt,:)=gcms3(:);
        gms(xcnt,:,:)=gcms3;
        xcnt=xcnt+1;
     end
  end

  for i=1:size(x,1)
     g=x(i,:);
     g=reshape(g,length(grt),length(gmz));
     tic(i,:)=sum(g');
  end
  figure; plot(tic');

  [u,s,v]=svd(tic,'econ');
  v1=v(:,1);
  figure; plot(grt,-(v1));

  save totalbinned_corrected_kerosene_nobaseline_nowavelet_rtincr=002 x grt gmz rtlev mzlev
-v7.3
```

```
end

%% BLP PLS
if task(3)==1
   load totalbinned_corrected_kerosene_nobaseline_nowavelet_rtincr=002

   if tic_is==1
      for i=1:size(x,1)
         temp=x(i,:);
         temp=reshape(temp,length(grt),length(gmz));
         tic(i,:)=sum(temp');
      end
      x=tic;
   else if tic_is==2
         for i=1:size(x,1)
            temp=x(i,:);
            temp=reshape(temp,length(grt),length(gmz));
            is(i,:)=sum(temp);
         end
         x=is;
      end
   end

   clear tic is

   for i=1:size(x,1)
      x(i,:)=x(i,:)./(norm(x(i,:)));
   end

   nboot=10;
   npart=3; %use npart = 3 to assure at least two samples in the training sets

%     for ii = 1:nboot
%        seqsave{ii} = latpart(y, npart);
%     end
%
%     save seqsave_kerosene.mat seqsave
   load seqsave_kerosene_AMDIS
   res_all=[];
   k=1;
   for ii = 1:nboot
      ii
      ypred_all_pls{ii}=0;
      seq=seqsave{ii};
      for jj = 1:npart
         tsq = seq(:, jj);
         ind = find(tsq > 0); % remove 0 padded values
         tsq = tsq(ind);
         x=full(x);
```

```
        [xt, xp, yt, yp] = makepart(tsq, x, y);

        %normalize to unit vector length
        for i=1:size(xt,1)
           xt(i,:)=xt(i,:)./norm(xt(i,:));
        end

        for i=1:size(xp,1)
           xp(i,:)=xp(i,:)./norm(xp(i,:));
        end

        [m,n]=size(xt);
        tic
        [u,s,v]=svd(xt,'econ');
        toc

        uxt=u*s;
        clear xt;
        uxp=xp*v;
        clear xp v;

        [res ypi]= opls_wl(uxt,yt,uxp, yp);
        ypred_all_plsi{ii,jj}=ypi;

        tic;   tree=mvid3(uxt,yt); toc
        yp_fures= uclass(tree, uxp);
        ypred_all_furesi{ii,jj}=yp_fures;

        pls_model=superpls(uxt,yt);
        ypred_all_superplsi{ii,jj}=plspred(pls_model,uxp);

        parameterCell{k} = {uxt,yt,uxp, yp};

        res=mgetres(uxt,yt);

        res_all((ii-1)*npart+jj,:)=res;
        resmean(k)=prod(res.^(1/length(res)));

        k=k+1;
     end
end

[u,s,v]=svd(x,'econ');
ux=u*s;
res_all=mgetres_all(ux,y);

save temp_result ypred_all_plsi ypred_all_furesi

%% analyze result
conf=zeros(nboot,size(uxp,1),size(uxp,1));
```

```
nboot=10;
npart=3; %use npart = 3 to assure at least two samples in the training sets
for ii = 1:nboot
   ii
   ypred_all_pls{ii}=0;
   seq=seqsave{ii};
   ncorrect_PLS(ii)=0;
   ncorrect_fures(ii)=0;
   ncorrect_superpls(ii)=0;
   for jj = 1:npart
      tsq = seq(:, jj);
      ind = find(tsq > 0); % remove 0 padded values
      tsq = tsq(ind);

      [xt, xp, yt, yp] = makepart(tsq, x, y);

      ypi=ypred_all_plsi{ii,jj};
      yp_fures=ypred_all_furesi{ii,jj};
      yp_superpls=ypred_all_superplsi{ii,jj};

      ncorrect_PLS(ii)=ncorrect_PLS(ii)+ypi;
      for ix=1:size(xp,1)
         %              if ypi(ix)==find(yp(ix,:));
         %                 ncorrect_PLS(ii)=ncorrect_PLS(ii)+1;
         %              end

         [temp, maxind]=max(yp_fures(ix,:));
         if maxind==find(yp(ix,:));
            ncorrect_fures(ii)=ncorrect_fures(ii)+1;
         end
         conf(ii,find(yp(ix,:)),maxind)=conf(ii,find(yp(ix,:)),maxind)+1;

         [temp, maxind]=max(yp_superpls(ix,:));
         if maxind==find(yp(ix,:));
            ncorrect_superpls(ii)=ncorrect_superpls(ii)+1;
         end

      end

   end
end
mean_fures=mean(ncorrect_fures./size(x,1));
mean_PLS=mean(ncorrect_PLS./size(x,1));
mean_superpls=mean(ncorrect_superpls./size(x,1));
mean_PDR=mean(resmean);
ci_fures=tinv(0.05, nboot-1, 2)*std(ncorrect_fures./size(x,1));
ci_PLS=tinv(0.05, nboot-1, 2)*std(ncorrect_PLS./size(x,1));
ci_superpls=tinv(0.05, nboot-1, 2)*std(ncorrect_superpls./size(x,1));
ci_PDR=tinv(0.05, nboot*npart-1, 2)*std(resmean);
fprintf('FuRES: %f %c %f\n',mean_fures*100,177,ci_fures*100);
```

```
    fprintf('oPLS: %f %c %f\n',mean_PLS*100,177,ci_PLS*100);
    fprintf('superpls: %f %c %f\n',mean_superpls*100,177,ci_superpls*100);
    fprintf('PDR: %f %c %f\n',mean_PDR,177,ci_PDR);

    misclassified_fures=mean(size(x,1)-ncorrect_fures);
    misclassified_PLS=mean(size(x,1)-ncorrect_PLS);
    ci_misclassified_fures=tinv(0.05, nboot-1, 2)*std(size(x,1)-ncorrect_fures);
    ci_misclassified_PLS=tinv(0.05, nboot-1, 2)*std(size(x,1)-ncorrect_PLS);
    fprintf('mis FuRES: %f %c %f\n',misclassified_fures,177,ci_misclassified_fures);
    fprintf('mis oPLS: %f %c %f\n',misclassified_PLS,177,ci_misclassified_PLS);


end

%% BLP PLS on component ratio
if task(4)==1
    load xratio_kerosene_new

    y=ysample;
    x=xratio;

    y=y(pickflag==1,:);
    y(:,sum(y)==0)=[];

    nboot=10;
    npart=3; %use npart = 3 to assure at least two samples in the training sets

    for ii = 1:nboot
       seqsave{ii} = latpart(y, npart);
    end

    save seqsave_kerosene_AMDIS.mat seqsave
    load seqsave_kerosene_AMDIS

    k=1;
    for ii = 1:nboot
       ii
       ypred_all_pls{ii}=0;
       seq=seqsave{ii};
       ncorrect_PLS(ii)=0;
       ncorrect_fures(ii)=0;
       for jj = 1:npart
          tsq = seq(:, jj);
          ind = find(tsq > 0); % remove 0 padded values
          tsq = tsq(ind);
          x=full(x);
          [xt, xp, yt, yp] = makepart(tsq, x, y);

          %autoscaling
          xt1=xt;
```

```
        xt=xt-repmat(mean(xt),size(xt,1),1);
        xt2=xt;
        xt=xt./repmat(std(xt),size(xt,1),1);
        xp=xp-repmat(mean(xt1),size(xp,1),1);
        xp=xp./repmat(std(xt2),size(xp,1),1);


        [m,n]=size(xt);

        [res ypi]= opls_wl(xt,yt,xp, yp);
        yt=double(yt);
        ypred_all_plsi{ii,jj}=ypi;

        tic;   tree=mvid3(uxt,yt); toc
        yp_fures= uclass(tree, uxp);
        ypred_all_furesi{ii,jj}=yp_fures;

        pls_model=superpls(xt,yt);
        ypred_all_superplsi{ii,jj}=plspred(pls_model,xp);

        res=mgetres(xt,yt);
        resmean(k)=prod(res.^(1/length(res)));

        k=k+1;

    end

end

save temp_result ypred_all_plsi ypred_all_furesi
%% analyze result
conf=zeros(nboot,size(xp,1),size(xp,1));
nboot=10;
npart=3; %use npart = 3 to assure at least two samples in the training sets
for ii = 1:nboot
    ii
    ypred_all_pls{ii}=0;
    seq=seqsave{ii};
    ncorrect_PLS(ii)=0;
    ncorrect_fures(ii)=0;
    ncorrect_superpls(ii)=0;
    for jj = 1:npart
        tsq = seq(:, jj);
        ind = find(tsq > 0); % remove 0 padded values
        tsq = tsq(ind);

        [xt, xp, yt, yp] = makepart(tsq, x, y);

        ypi=ypred_all_plsi{ii,jj};
        yp_fures=ypred_all_furesi{ii,jj};
```

```
        yp_superpls=ypred_all_superplsi{ii,jj};

        ncorrect_PLS(ii)=ncorrect_PLS(ii)+ypi;
        for ix=1:size(xp,1)


           [temp, maxind]=max(yp_fures(ix,:));
           if maxind==find(yp(ix,:));
              ncorrect_fures(ii)=ncorrect_fures(ii)+1;
           end
           conf(ii,find(yp(ix,:)),maxind)=conf(ii,find(yp(ix,:)),maxind)+1;

           [temp, maxind]=max(yp_superpls(ix,:));
           if maxind==find(yp(ix,:));
              ncorrect_superpls(ii)=ncorrect_superpls(ii)+1;
           end

        end

     end
  end
  mean_fures=mean(ncorrect_fures./size(x,1));
  mean_PLS=mean(ncorrect_PLS./size(x,1));
  mean_superpls=mean(ncorrect_superpls./size(x,1));
  mean_PDR=mean(resmean);
  ci_fures=tinv(0.05, nboot-1, 2)*std(ncorrect_fures./size(x,1));
  ci_PLS=tinv(0.05, nboot-1, 2)*std(ncorrect_PLS./size(x,1));
  ci_superpls=tinv(0.05, nboot-1, 2)*std(ncorrect_superpls./size(x,1));
  ci_PDR=tinv(0.05, nboot*npart-1, 2)*std(resmean);
  fprintf('FuRES: %f %c %f\n',mean_fures*100,177,ci_fures*100);
  fprintf('oPLS: %f %c %f\n',mean_PLS*100,177,ci_PLS*100);
  fprintf('superpls: %f %c %f\n',mean_superpls*100,177,ci_superpls*100);
  fprintf('PDR: %f %c %f\n',mean_PDR,177,ci_PDR);

  misclassified_fures=mean(size(x,1)-ncorrect_fures);
  misclassified_PLS=mean(size(x,1)-ncorrect_PLS);
  ci_misclassified_fures=tinv(0.05, nboot-1, 2)*std(size(x,1)-ncorrect_fures);
  ci_misclassified_PLS=tinv(0.05, nboot-1, 2)*std(size(x,1)-ncorrect_PLS);
  fprintf('mis FuRES: %f %c %f\n',misclassified_fures,177,ci_misclassified_fures);
  fprintf('mis oPLS: %f %c %f\n',misclassified_PLS,177,ci_misclassified_PLS);


end

%% plot PDR and classification error
% res_mean=mean(res_all);
figure
res_grid=[];
icnt=1;
```

```
for i=1:(size(y,2)-1)
   for j=(i+1):size(y,2)

%        res_grid(j,i)=sum(res_all(:,icnt)<1.5);
       res_grid(j,i)=prod(res_all(:,icnt).^(1/9));
%        res_grid(j,i)=min(res_all(:,icnt));
       if res_grid(j,i)>=5 %limit PDR to a maximum of 3
          res_grid(j,i)=5;
       end
       icnt=icnt+1;

   end
end

% res_grid(j,i)=9; %fake a grid to get the desired scale
%
% fig1=imagesc(res_grid);
% set(gca,'TickLength',[0
0.025],'YDir','reverse','Layer','top','CLim',[0,max(max(log(res_grid)))])
set(gca,'TickLength',[0 0.025],'YDir','reverse','Layer','top','CLim',[0,max(max(res_grid))])
% set(gca,'TickLength',[0 0.025],'YDir','reverse','Layer','top','CLim',[0,9])
% Colormap([1 1 1;0.98412698507309 0.98412698507309
0.98412698507309;0.968253970146179 0.968253970146179
0.968253970146179;0.952380955219269 0.952380955219269
0.952380955219269;0.936507940292358 0.936507940292358
0.936507940292358;0.920634925365448 0.920634925365448
0.920634925365448;0.904761910438538 0.904761910438538
0.904761910438538;0.888888895511627 0.888888895511627
0.888888895511627;0.873015880584717 0.873015880584717
0.873015880584717;0.857142865657806 0.857142865657806
0.857142865657806;0.841269850730896 0.841269850730896
0.841269850730896;0.825396835803986 0.825396835803986
0.825396835803986;0.809523820877075 0.809523820877075
0.809523820877075;0.793650805950165 0.793650805950165
0.793650805950165;0.777777791023254 0.777777791023254
0.777777791023254;0.761904776096344 0.761904776096344
0.761904776096344;0.746031761169434 0.746031761169434
0.746031761169434;0.730158746242523 0.730158746242523
0.730158746242523;0.714285731315613 0.714285731315613
0.714285731315613;0.698412716388702 0.698412716388702
0.698412716388702;0.682539701461792 0.682539701461792
0.682539701461792;0.666666686534882 0.666666686534882
0.666666686534882;0.650793671607971 0.650793671607971
0.650793671607971;0.634920656681061 0.634920656681061
0.634920656681061;0.61904764175415 0.61904764175415
0.61904764175415;0.60317462682724 0.60317462682724
0.60317462682724;0.58730161190033 0.58730161190033
0.58730161190033;0.571428596973419 0.571428596973419
0.571428596973419;0.555555582046509 0.555555582046509
0.555555582046509;0.539682567119598 0.539682567119598
```

```
0.539682567119598;0.523809552192688 0.523809552192688
0.523809552192688;0.507936537265778 0.507936537265778
0.507936537265778;0.492063492536545 0.492063492536545
0.492063492536545;0.476190477609634 0.476190477609634
0.476190477609634;0.460317462682724 0.460317462682724
0.460317462682724;0.444444447755814 0.444444447755814
0.444444447755814;0.428571432828903 0.428571432828903
0.428571432828903;0.412698417901993 0.412698417901993
0.412698417901993;0.396825402975082 0.396825402975082
0.396825402975082;0.380952388048172 0.380952388048172
0.380952388048172;0.365079373121262 0.365079373121262
0.365079373121262;0.349206358194351 0.349206358194351
0.349206358194351;0.333333343267441 0.333333343267441
0.333333343267441;0.31746032834053 0.31746032834053
0.31746032834053;0.30158731341362 0.30158731341362
0.30158731341362;0.28571429848671 0.28571429848671
0.28571429848671;0.269841283559799 0.269841283559799
0.269841283559799;0.253968268632889 0.253968268632889
0.253968268632889;0.238095238804817 0.238095238804817
0.238095238804817;0.222222223877907 0.222222223877907
0.222222223877907;0.206349208950996 0.206349208950996
0.206349208950996;0.190476194024086 0.190476194024086
0.190476194024086;0.174603179097176 0.174603179097176
0.174603179097176;0.158730164170265 0.158730164170265
0.158730164170265;0.142857149243355 0.142857149243355
0.142857149243355;0.126984134316444 0.126984134316444
0.126984134316444;0.111111111938953 0.111111111938953
0.111111111938953;0.095238097012043 0.095238097012043
0.095238097012043;0.0793650820851326 0.0793650820851326
0.0793650820851326;0.0634920671582222 0.0634920671582222
0.0634920671582222;0.0476190485060215 0.0476190485060215
0.0476190485060215;0.0317460335791111 0.0317460335791111
0.0317460335791111;0.0158730167895556 0.0158730167895556 0.0158730167895556;0 0 0]);
colormap('gray')
text(13.15,1.5,'\geq')
% colormap(flipud(colormap))
colorbar
% tickp=[0 1.5 10 50 100 200];

% colorbar('YLim', [0,max(max(log(res_grid)))],'Ytick', log(tickp), 'Yticklabel', num2str(tickp'));

colorbar('Yticklabel', {'0';'0.5';'1';'1.5';'2';'2.5';'3';'3.5';'4';'4.5';'  5'});
box off

res1=(res_grid);
res1(isinf(res1))=inf;
minres=min(min(res1));

fures_grid=(squeeze(sum(conf)));
```

```
for i=1:(size(yt,2)-1)
    fures_grid(i,i)=0;
   for j=(i+1):size(yt,2)

       fures_grid(j,i)=fures_grid(j,i)+fures_grid(i,j);
       fures_grid(i,j)=0;
       icnt=icnt+1;
       color_code=((res_grid(j,i))-minres)/max(max((res_grid))-minres);
       rectangle('Position',[i-0.5,j-0.5,1,1],'FaceColor',[color_code,color_code,color_code]);
       if fures_grid(j,i)<1.5

       end
       if fures_grid(j,i)>0
          if res_grid(j,i)>2.5

text(i,j,num2str(fures_grid(j,i)),'HorizontalAlignment','center','VerticalAlignment','middle','color'
,'black');
          else

text(i,j,num2str(fures_grid(j,i)),'HorizontalAlignment','center','VerticalAlignment','middle','color'
,'white');
          end
       end
   end
end

% rectangle('Position',[i-0.5,j-0.5,1,1],'FaceColor','White'); %cover faked grid

ylim([1.5, 13.5]);
xlim([0.5, 12.5]);

xlabel('Sample ID')
ylabel('Sample ID')
```

Functions called (in alphabetical order)

LATPART

```
function tpart=latpart(y, part)
% tpart=latpart(Y, part)
% returns sequence numbers for test sets with each column in a partition
% empty sequences are coded as zeros
% assumes class i.d.'s are encoded as 1s and 0s in Y

[my, ny] = size(y);
nclass = sum(y);

nlen = ceil(my/part);

tpart = [];
for i=1:ny
   ind = find(y(:,i));
   mind = length(ind);
   tpart = [tpart; ind(randperm(mind))];
end;
npad = part*nlen-my;
tpart = [tpart; zeros(npad, 1)];

tpart = reshape(tpart, part, nlen)';

% npart = ceil(nclass/part);
% nlen = sum(npart);
% tpart = zeros(nlen, part);
%
% cnt = 1;
% for i=1:ny
%    [mind, nind] = find(y(:,i)==max(y(:,i)));
%    rn = rand(length(mind), 1);
%    [t, idx] = sort(rn);
%    ik = 1;
%    for j=1:part;
%       icnt = cnt;
%       for k=1:npart(i);
%          if ik <= length(mind)
%             tpart(icnt, j) = mind(idx(ik));
%          else
%             tpart(icnt, j) = 0;
%          end
%          ik=ik+1;
%          icnt=icnt+1;
%       end
%    end
%    cnt = icnt;
% end
```

MAKEPART

```
function [xt, xp, yt, yp] = makepart(seq, x, y)
% [xt, xp, yt, yp] = makepart(seq, x, y)
% given the test sequence encoded as a column vector splits the x and y
% data into two sets based on the partition.

[m, n] = size(y);

nseq = setdiff([1:m], seq);

xp = x(seq, :);
yp = y(seq, :);

xt = x(nseq, :);
yt = y(nseq, :);
```

MGETRES

```
function [res indr indc res_weighed]= mgetres(x, y)
% res = mgetres(x, y)
% perform multivariate resolution by generating a set of projections onto the
% difference vector of two class averages
% between pairs of means divided by 2 times there summed standard
% deviations.

[my, ny] = size(y);

icnt = 1;
for ii = 1:(ny-1)
   aind = find(y(:, ii));
   for jj = (ii+1):ny
      bind = find(y(:, jj));
      res(icnt, :) = getresdiff(x(aind, :), x(bind, :));
      indr(icnt, :)= ii;
      indc(icnt, :)= jj;
      res_weighed(icnt, :)= res(icnt, :).*(length(aind)+length(bind));
      icnt = icnt+1;
   end
end
%*************************************************************************
function res = getresdiff(a, b)
% uses projected difference method to calculate the resolution between two
% groups of data.  Spectra are stored as rows

% ma = mean(a);
% mb = mean(b);

%function inline for speedups
```

```
%Weiying Lu

na=size(a,1);
nb=size(b,1);

% ma=sum(a)/na;
% mb=sum(b)/nb;

diff = (sum(a)/na-sum(b)/nb)';

pa = a*diff;
pb = b*diff;

ma = sum(pa)/na;
mb = sum(pb)/nb;

% sa = std(pa);
% sb = std(pb);

sa = sqrt(sum((pa-ma).^2)./na);
sb = sqrt(sum((pb-mb).^2)./nb);

res = abs(ma-mb)/(2*(sa+sb));



MGETRES_ALL
function res_all=mgetres_all(x,y)
%calculates all possible combination of PDRs when the number of objects in
%each class is small enough(<5).
[my, ny] = size(y);

icnt = 1;

for ii = 1:(ny-1)
   aind = find(y(:, ii));
   for jj = (ii+1):ny
      bind = find(y(:, jj));

      for iboot=1:length(aind)
         for jboot=1:length(bind)

            aind1=aind; bind1=bind;
            aind1(iboot)=[];bind1(jboot)=[];
            x11=x(aind1,:);x12=x(bind1,:);
            x1=[x11;x12];
            y1=zeros(size(x1,1),2);
            y1(1:length(aind1),1)=1;
            y1(length(aind1)+(1:length(bind1)),2)=1;
```

```
            res_all((iboot-1)*length(aind)+jboot,icnt)=mgetres(x1,y1);
          end
        end

        icnt = icnt+1;
    end
end
```

MVID3

```
function node = mvid3(x, y, maxent1, maxrule1)
% node = mvid3(x, y, [maxent], [maxrule])
% node is a node of a classification tree
% x is a matrix of an independent variables with objects as rows
%
% NOTE:  the x matrix last column must be all values of 1 to model the
% intercept
%
% y is a binary matrix with class designees in the the columns (only one
% class per object is supported
%
% maxent is a pruning parameter the default is 0; if a rule's absolute
% entropy is below this threshold it will be pruned, 0 is no pruning
%
% maxrule is a pruning parameter that sets the maximum number of rules
% the default is inf
%
% Author Peter Harrington@Ohio.edu Version 4.0
% Harrington, P. D. Minimal Neural Networks - Differentiation of Classification Entropy.
Chemometrics and Intelligent Laboratory Systems 1993, 19, 143-154.
% P.B. Harrington, Chemometr. Intell. Lab. Syst. 19 (1993) 143-154.
% FuRES wrapper function
%

global X Y mG maxent maxrule;
if nargin < 4
   maxrule = inf;
else
   maxrule = maxrule1;
end;

if nargin < 3
   maxent = 0;
else
   maxent = maxent1;
end;

[m, n] = size(y);
poss = ones(m, 1);
x = [x, ones(m, 1)];
```

```
X = x;
Y = y;
mG = size(X, 1); % global m-value

node = [];
node.ny = n;
node = mv_id3(node, poss);
%*************************************************************************
function node = mv_id3(node, poss)
% node = mv_id3(node, poss)
% node is a node of a classification tree
% x is a matrix of an independent variables with objects as rows
%
% NOTE:  the x matrix last column must be all values of 1 to model the
% intercept
%
% y is a binary matrix with class designees in the the columns (only one
% class per object is supported
%
%
% Author Peter Harrington@Ohio.edu Version 1.0
% P.B. Harrington, Chemometr. Intell. Lab. Syst. 19 (1993) 143-154.
global X Y maxent;

ind = find(poss > 0.5);
y = Y(ind, :);
x = X(ind, :);

ynum = sum(max(y, [], 1));
if ynum == 1; % only one class nothing to do here
   node.cat = find(max(y, [], 1));
   node.numleft = size(y, 1);
   node.ids = ind;
   return;
elseif ynum == 0; % bad point can't get rid of it
   [t, ti] = max(poss);
   node.cat = find(Y(ti, :));
   node.numleft = size(y, 1);
   node.ids = ind;
   return;
end

% search for partitions
% repeat = 10;
% for i = 1:repeat;
%     [mh(i), mtemp(i), mwt(:, i)] = mvfuzzyentopt(ind);
% end;
% %
% % [mx, mi] = max(max(abs(mwt(1:(end-1), :))));
```

```
% % [mx, mi] = min(mh);
% % [mx, mi] = max(mtemp);
% [mx, mi] = min(mh./mtemp);
% %
% h = mh(mi);
% temp = mtemp(mi);
% wt = mwt(:, mi);
%
[h, temp, wt] = mvfuzzyentopt(ind);
%
% set the tree up so that the tree is defined from left to right

if h > maxent
   dots = x*wt;
   ind = find(dots < 0);
   [t, tia] = max(sum(y(ind, :)));
   ind = find(dots > 0);
   [t, tib] = max(sum(y(ind, :)));

   if tia > tib
      wt = -wt;
   end

   dots = 1./(1+exp(-(X*wt)./temp));

   node.h = h;
   node.wt = wt;
   node.t = temp;
   node.cat = 0;

   node.left=[];
   % node.left = mv_id3(node.left, (1-dots).*poss);
   node.left = mv_id3(node.left, min((1-dots), poss));

   node.right=[];
   % node.right = mv_id3(node.right, dots.*poss);
   node.right = mv_id3(node.right, min(dots, poss));
else
   [s, node.cat] = max(sum(y));
   node.numleft = size(y, 1);
   node.ids = ind;
   return;
end;

%***************************************************************************
function [h, temp, wt] = mvfuzzyentopt(ind)
% [h, temp, wt] = mvfuzzyentopt(x, y)
% x is a matrix of an independent variables with objects as rows
%
% NOTE:  the x matrix last column must be all values of 1 to model the
```

```
% intercept
%
% y is a binary matrix with class designees in the the columns (only one
% class per object is supported
%
% h is the entropy of classification
% temp is temperature
%  wt is a wt vector similar to an ANN unit.  The first n-1 components have
% been normalized to unit length
% wt(n) is the bias value
%
% Author Peter Harrington@Ohio.edu Version 1.0
% P.B. Harrington, Chemometr. Intell. Lab. Syst. 19 (1993) 143-154.

global X Y TEMP NET XSub YSub;

YSub = Y(ind, :);
XSub = X(ind, :);
EPS = 2E-8;
ZEPS = 1E-12;

warning off;

[m, n] = size(XSub);
n1 = n-1;

% find the largest separation defined by resolution among all class
% averages and develop a discriminant

 ip = randperm(m);
 wt = (XSub(ip(1),:)-XSub(ip(2),:))';

% wt = mean(x(:, 1:n1))';
den = sqrt(sum(wt(1:n1).^2));
wt = wt/den;

dots = XSub*wt;
wt(n) = -mean(dots);
temp = 10*max(max(dots+wt(n)), -min(dots+wt(n)));
options = [];

oh = 1;
ot=0;
h = 1;
h1 = 1;
% options = optimset('MaxIter', 1e6, 'MaxFunEvals', 1e6, 'Diagnostics', 'off');
TEMP = temp;
opts = optimset('MaxFunEvals', 1e8, 'MaxIter', 1e16, 'Diagnostics', 'on', 'TolFun', 1e-
6,'TolX',1e3);
```

```
[wt, h,exitflag, output] = fminsearch(@mvfuzzyent, wt, opts);
for i = 1:100
   TEMP = temp;
   ot = temp;
   oh = h;
%     wt(1:n1) = sign(wt(1:n1)).*wt(1:n1).^2;
%     den = sqrt(sum(wt(1:n1).^2));
%     wt(1:n1) = wt(1:n1)/den;
   opts = optimset('MaxFunEvals', 1e8, 'MaxIter', 1e16, 'Diagnostics', 'off', 'TolFun', 1e-
6,'TolX',1e-6);
   [wt, h] = fminunc(@mvfuzzyent, wt, opts);
   den = sqrt(sum(wt(1:n1).^2));
   wt(1:n1) = wt(1:n1)/den;

%     test(:, i) = wt(1:n1);
   NET = XSub*wt;
   minNet = min(NET);
   maxNet = max(NET);
   if minNet < 0 && maxNet > 0
      [temp, h1] = fminbnd(@fuzzyentdert, 0, max(abs(NET)));
   else
      opts = optimset('MaxFunEvals', 1e8, 'MaxIter', 1e16, 'Diagnostics', 'off', 'TolFun', 1e-
6,'TolX',1e3);
      [wt, h] = fminsearch(@mvfuzzyent, wt, opts);
      temp = temp/1.1;
   end

   [i, h, h1, temp]
   if abs(ot-temp) < EPS*temp+ZEPS;
      break;
   end
end
%*********************************************************************************
function h = mvfuzzyent(wt)
% h = mvfuzzyent(wt)
% wt is a wt vector similar to an ANN unit.  The first n-1 components have
% been normalized to unit length
% wt(n) is the bias value
% h is the entropy of classication
% internal global variables are X Y TEMP and NET
% returns the entropy of classification and attribute for data x and y
%
% Author Peter Harrington@Ohio.edu Version 1.0
% P.B. Harrington, Chemometr. Intell. Lab. Syst. 19 (1993) 143-154.
%
% warnings are turned off to avoid logs of zero

global XSub YSub TEMP mG;

n = length(wt)-1;
```

```
den = sum(wt(1:n).^2);
den = sqrt(den);
% den = sum(abs(wt(1:n)));
wt(1:n) = wt(1:n)/den;

xt = XSub*wt;
xt = 1./(1+exp(-xt/TEMP));
% calculate postive entropies

yp = xt'*YSub;
yps = sum(yp);
yp = yp/yps; % now we have probabilities for each class
hp = -yp.*log(yp);
ind = find(isfinite(hp));
hpc = sum(hp(ind));

% now the negative case
xt = 1-xt;
yn = xt'*YSub;
yns = sum(yn);
yn = yn/yns;
hn = -yn.*log(yn);
ind = find(isfinite(hn));
hnc = sum(hn(ind));

h = (yps*hpc + yns*hnc)/mG;
%*************************************************************************
function h1 = fuzzyentdert(temp)
% returns the first derivative of the entropy of classification with respect to temp for data x and y
% global Y can't contain columns of zeros
% Requires global NET is defined as X-ATTRIB
% internal global variables are NET and Y
% Author Peter Harrington@Ohio.edu Version 2.0
% P.B. Harrington, Chemometr. Intell. Lab. Syst. 19 (1993) 143-154.

global NET YSub mG;

if temp==0;
   h1 = 0;
   return;
end;

xt = 1./(1+exp(-NET/temp));
xt1 = -xt.*(1-xt).*NET/temp.^2;

% calculate postive entropies

yp = xt'*YSub;
yps = sum(yp);
yp = yp/yps; % now we have probabilities for each class
```

```
hp = -yp.*log(yp);
ind = find(isfinite(hp));
hpc = sum(hp(ind));

yp1 = xt1'*YSub; % now take care of the derivatives
yp1s = sum(yp1);
yp1 =  yp1/yps - yp*sum(xt1)/yps^2;
hp1 = -yp1.*(log(yp)+1);
ind = find(isfinite(hp1));
hp1c = sum(hp1(ind));

% now the negative case
xt = 1-xt;
yn = xt'*YSub;
yns = sum(yn);
yn = yn/yns;
hn = -yn.*log(yn);
ind = find(isfinite(hn));
hnc = sum(hn(ind));

xt1 = -xt1;
yn1 = xt1'*YSub; % now take care of the derivatives
yn1s = sum(yn1);
yn1 = yn1/yns - yn*sum(xt1)/yns^2;
hn1 = -yn1.*(log(yn)+1);
ind = find(isfinite(hn1));
hn1c = sum(hn1(ind));

% h = (yps*hpc + yns*hnc)/m;
h1 = -(yp1s*hpc + yps*hp1c + yn1s*hnc + yns*hn1c)/mG;
```

OPLS_W1

```
function [yp ypi]=opls_wl(xt, yt, xp, yp)
%Optimal partial least squares discriminant analysis (oPLS-DA)  training
%and prediction, NIPALS PLS2 algorithm
%in one function so it will run faster
%
%xt,yt --x and y matrix for training
%xp   --x matrix for prediction
%yp --the prediction result by number of corrected prediction
%
%code derived from pls.m our_pls.m and pls_pred.m

MAXITER = 1000;
EPS = 1E-6;

% center X block and y block;
meanx = mean(xt,1);
mx = size(xt,1);
x = xt-ones(mx ,1)*meanx;

meany = mean(yt,1);
my = size(yt,1);
y = yt-ones(my,1)*meany;

m=size(xp,1);
xp = xp-ones(m,1)*meanx;
yp_temp = ones(m,1)*meany;
[rowresult, ypc]=find(yp==1);

ypc(rowresult)=ypc;
ncomp=max(size(xt));
yp=zeros(1,ncomp);

for ii=1:ncomp

   [s, is] = max([sum(y.^2), 0]);
   tnew  = y(:, is);
   for j = 1:MAXITER
      % xblock
      w = tnew'*x/sum(tnew.^2);
      w = w/norm(w);
      t = x*w';

      % yblock
      q = t'*y/(t'*t);
      %        q = q/norm(q);
      %        u = y*q'/norm(q);
      told = tnew;
      tnew = t;
      if sum((told-tnew).^2) < EPS*length(t)
```

```
            break;
        end
    end

    p = t'*x/sum(t.^2);

    x = x - t*p;
    y  = y - t*q;

    %predict xp
    tp = xp * w';
    xp = xp-tp*p;
    yp_temp = yp_temp + tp * q;

    %find prediction classes
    [temp,ind]=max(yp_temp,[],2);

    %number of correct prediction objects
    resultc=sum(ind==ypc);

    %store result for each latent variable number
    yp(ii) = resultc;
end;

% %the prediction result by using all latent variables
% ypi=ind;
ypi=max(yp);


OUR_PLS
function pls = our_pls(x, y, ncomp, mode)
% pls = our_pls(x, y, ncomponents, mode);
% mode = 'bestclass' terminates the model when 100% classification is
% achieved.  Asssumes binary encoding and max classification.

if nargin < 4;
    mode = '';
end;

mode = lower(mode);

MAXITER = 1000;
EPS = 1E-12;

[mx, nx] = size(x);
[my, ny] = size(y);

if mx ~= my
    disp(sprintf('Error: Number of rows of X and Y are not equal:  %u, and %u \a', mx, my));
    pls = 0;
```

```
   return;
end

if nargin == 2
   ncomp = min(mx, nx);
end

minvar = min([mx, nx, ncomp]);

meanx = mean(x);
meany = mean(y);

if strcmp(mode, 'bestclass');
   yp = ones(my, 1)*meany;
   [s, iclass] = max(y, [], 2);
end;

x = x - ones(mx, 1)*meanx;
y = y - ones(my, 1)*meany;

pls.meanx = meanx;
pls.meany = meany;

for ii = 1:minvar;
   [s, is] = max([sum(y.^2), 0]);
   tnew  = y(:, is);
   for j = 1:MAXITER
      % xblock
      w = tnew'*x/(tnew'*tnew);
      w = w/norm(w);
      t = x*w';

      % yblock
      q = t'*y/(t'*t);
      %        q = q/norm(q);
      u = y*q'/norm(q);
      told = tnew;
      tnew = t;
      if sum((told-tnew).^2) < EPS
         break;
      end
   end

   p = t'*x/(t'*t);

   %    plen = norm(p);
   %    p = p/plen;
   %    t = t*plen;
   %    w = w*plen;
   %    b = u'*t/(t'*t);
```

```
   x = x - t*p;
   y  = y - t*q;
   pls.w{ii} = w';
   pls.p{ii} = p;
   pls.q{ii} = q;

   if strcmp(mode, 'bestclass');
      yp = yp + t*q;
      [t, it] = max(yp, [], 2);
      score = sum(abs(iclass - it));
      if score == 0;
         break;
      end
   end
   if max(max(abs(x))) < EPS;
      break;
   end
   if max(max(abs(y))) < EPS;
      break
   end
end
pls.ncomp = ii;
% save variables: pls.meanx
%               pls.meany
%               pls.w
%               pls.p
%               pls.q
%               pls.b
%pls
PLSPRED
function ye = plspred(p, x, ncomp)
% yp = plspred(p, x)

% returns the predictions that best fit the targe with respect to latent
% variable number
if nargin == 2
   ncomp = p.ncomp;
end

[m, n] = size(x);
x = x - ones(m, 1)*p.meanx;
ye = ones(m,1)*p.meany;

for ii = 1:ncomp
   t = x*p.w{ii};
   x = x-t*p.p{ii};
   ye = ye + t*p.q{ii};
end
```

SMARTBASELINE2

```
function gcms = smartbaseline2(gcms, numend, numbasis, err)
% version 1 performs the exact calculation for the most negative corrected
% peak in the mass spectrum
% err is negative and in absolute units
% gcms = smartbaseline1(x, numbasis, numend)
% regularized gram schmidt subtracts background until a fixed negative
% error is achieved

if max(size(numend)) == 1
   backgrounds = gcms(end-numend+1:end, :);
else
   backgrounds = gcms(numend, :);
end

[v, s, u] = svd(backgrounds', 0);

[m, n] = size(gcms);

for i = 1:m
   ms = gcms(i, :);
   ems = ms*v(:, 1:numbasis)*v(:, 1:numbasis)';
   tms =  ms - ems;
   ii = find(ems > 0);
   [s, is] = min((ms(ii)-err)./ems(ii));
   s = ms(ii(is)) - ems(ii(is));
   if s < err
      lambda  = (ms(ii(is))-err)/ems(ii(is));
      tms = ms - lambda*ms*v(:, 1:numbasis)*v(:, 1:numbasis)';
% apply the correction
   end
%       tms(tms<0)=0;
   gcms(i, :) = tms;
end
```

SUPERPLS

```
function gcms = smartbaseline2(gcms, numend, numbasis, err)
% version 1 performs the exact calculation for the most negative corrected
% peak in the mass spectrum
% err is negative and in absolute units
% gcms = smartbaseline1(x, numbasis, numend)
% regularized gram schmidt subtracts background until a fixed negative
% error is achieved

if max(size(numend)) == 1
   backgrounds = gcms(end-numend+1:end, :);
else
   backgrounds = gcms(numend, :);
end

[v, s, u] = svd(backgrounds', 0);

[m, n] = size(gcms);

for i = 1:m
   ms = gcms(i, :);
   ems = ms*v(:, 1:numbasis)*v(:, 1:numbasis)';
   tms =  ms - ems;
   ii = find(ems > 0);
   [s, is] = min((ms(ii)-err)./ems(ii));
   s = ms(ii(is)) - ems(ii(is));
   if s < err
     lambda  = (ms(ii(is))-err)/ems(ii(is));
     tms = ms - lambda*ms*v(:, 1:numbasis)*v(:, 1:numbasis)';
% apply the correction
   end
%       tms(tms<0)=0;
   gcms(i, :) = tms;
end
```

TDIST

```
function q = tdist(t, df, sided)
% q = tdist(t, df, sided)
% returns probability alpha
% given a t-statistic
% df=degrees of freedom
% sided specifies the sides to the distribution 1 or 2
if df == 0
   disp('Error:  Zero Degrees of Freedom\a');
   g=-1;
   return;
end;
q=0.5*betainc(df/(df+t.^2), df/2, 0.5)*sided;
```

UCLASS

```
function [yp id] = uclass(node, x)
% yp = uclass(node, x)
% fuzzy classification algorithm used for both FURES and Fuzzy ID3
%
% node is a node of a classification tree
% x is a matrix of an independent variables with objects as rows
%
% Wrapper function
%
% poss is a column vector of ones initially and represents the cumulative
% possibility or fuzziness for each object at each node in the tree
%
% yp is the predicted y matrix and is defined as the maximum possibility
% for each class
%
% y is a binary matrix with class designees in the the columns (only one
% class per object is supported
%
%
% Author Peter Harrington@Ohio.edu Version 5.0
% P.B. Harrington, Chemometr. Intell. Lab. Syst. 19 (1993) 143-154.
global m;
[m, n] = size(x);

poss = ones(m, 1);
n=node.ny;

yp = zeros(m, n);
% augment x


% if length(node.wt) > 1
%     x = [x, ones(m, 1)];
% end

yp = u_class(node, x, poss, yp);
%*************************************************************************
function yp = u_class(node, x, poss, yp)
% yp = uclass(node, x, poss, yp)
% fuzzy classification algorithm used for both FURES and Fuzzy ID3
%
% node is a node of a classification tree
% x is a matrix of an independent variables with objects as rows
%
% NOTE:  the x matrix last column must be all values of 1 to model the
% intercept
%
% poss is a column vector of ones initially and represents the cumulative
% possibility or fuzziness for each object at each node in the tree
```

```
%
% yp is the predicted y matrix and is defined as the maximum possibility
% for each class
%
% y is a binary matrix with class designees in the the columns (only one
% class per object is supported
%
%
% Author Peter Harrington@Ohio.edu Version 3.0
% P.B. Harrington, Chemometr. Intell. Lab. Syst. 19 (1993) 143-154.
global m;

if node.cat > 0
   yp(:, node.cat) = max(yp(:, node.cat), poss);
   return
end

if length(node.wt)==1
   xt=(x(:,node.i)-node.wt)/node.t;
else
   if isfield(node, 'v')
      ux = x - ones(m, 1)*node.mx;
      ux = ux*node.v;
      xt = [ux ones(m, 1)]*node.wt/node.t;
   elseif isfield(node, 'mx')
      ux = x - ones(m, 1)*node.mx(1:(end-1));
      xt = [ux ones(m, 1)]*node.wt/node.t;
   else
      xt = [x ones(m, 1)]*node.wt/node.t;
   end
end
xt = 1./(1+exp(-xt));

yp = u_class(node.left, x, min((1-xt), poss), yp);
yp = u_class(node.right, x, min(xt, poss), yp);

% yp = u_class(node.left, x, (1-xt).*poss, yp);
% yp = u_class(node.right, x, xt.*poss, yp);
```

Appendix 4

R Scripts

Main Routines

```
#Correlation_scores_with_roc-analysis.R

library(caret)
library(lattice)
library(ROCR)
source("/Users/npetraco/codes/R/chemometric_utilities/sourceme.R")

#Load spread sheet of ratios:
#File name components:
rootd<-"/Users/npetraco/latex/papers/graham_gas/data/"
raw<-read.csv("/Users/npetraco/latex/papers/graham_gas/data/BigMPD.csv",header=TRUE)

#Name labels for each type of MPD:
lbl<-raw[,1]
lbl.real<-as.character(raw[,2])
raw2<-raw[,3:ncol(raw)]
colnames(raw2)
raw3<-clean.up.spreadsheet(raw2)

#Indices of problem spectra:
idx.prob<-as.numeric(sapply(c(121,123,132,135,149,150,152,155),function(x){which(lbl==x)}))
lbl<-lbl[-idx.prob]
X<-raw3[-idx.prob,]
dim(X)

Xs<-scale(X,center=TRUE,scale=TRUE)[,]
score.info<-correlation.compare3(Xs, lbl, lbl.real, cor.typ="kendall", sames.warning.cutoff=0.5,
difs.warning.cutoff=0.501, outfil.path=NULL)

#Score Histograms
hist(score.info[[1]],freq=FALSE,col=rgb(0,1,0,1/4),ylim=c(0,4),xlim=c(-
0.6,1),ylab="",xlab="",main="")
lines(density(score.info[[1]],width=0.15))
par(new=TRUE)
hist(score.info[[2]],freq=FALSE,col=rgb(1,0,0,1/4),ylim=c(0,4),xlim=c(-0.6,1),xlab="Kendall-
tau",main="Kendall-tau Scores between MPD Peak Ratios")
lines(density(score.info[[2]],width=0.5))

#Stack columns to make box-whiskers plots
same.lbl<-rep("same",length(score.info[[1]]))
diff.lbl<-rep("different",length(score.info[[2]]))
sd.lbl<-factor(c(same.lbl,diff.lbl))
all.scores<-c(score.info[[1]],score.info[[2]])
score.tab<-data.frame(all.scores,sd.lbl)
```

```
bwplot(all.scores~sd.lbl,data=score.tab)

#ROC curve
pred<-prediction(all.scores,sd.lbl)
                    #y     #x
perf<-performance(pred,"tpr","fpr")
tpr.vals<-perf@y.values[[1]]
fpr.vals<-perf@x.values[[1]]
cutoffs<-perf@alpha.values[[1]]

roc.zoom<-cbind(cutoffs,100*tpr.vals,100*fpr.vals)
colnames(roc.zoom)<-c("cutoffs","TPR","FPR")
#roc.zoom
low<-0
high<-5.1
roc.zoom.plot<-roc.zoom[(roc.zoom[,3]<=high & roc.zoom[,3]>=low),]
plot(perf)
plot(roc.zoom.plot[,3],roc.zoom.plot[,2],typ="l",main="ROC Curve",xlab="False Positive Rate
(%)",ylab="True Positive Rate (%)")
roc.zoom.plot

#pearson 0.5279934
#tpr: 91.8279570
#fpr: 5.000232742

#kendall 0.5037879
#tpr: 98.279570
#fpr: 5.063538612
#Average pairwise decision error rate: 4.94531

#spearman: 0.5989305
#tpr: 96.129032
#fpr: 5.001163711

#GET AVERAGE OVERALL ERROR RATE AS WELL**********
thresh<-0.5037879
sames<-score.info[[1]]
diffs<-score.info[[2]]
num.fp<-length(sames)-length(which(sames>thresh))
num.fn<-length(diffs)-length(which(diffs<thresh))
#percentage of decisions that are wrong at the threshold:
(num.fp+num.fn)/(length(sames)+length(diffs))*100
#FNR:
100-98.279570 #ROCR
num.fp/length(sames)*100 #Counts by thresh
#FPR:
5.063538612 #ROCR
num.fn/length(diffs)*100 #Counts by thresh
```

```
#Intergroup distances as defined by pdr:
dinfo<-pdr.dists(Xs,as.factor(lbl))
which(dinfo[,3]=="NaN")
dinfo[which(dinfo[,3]=="NaN"),]
pdr.dist(Xs,lbl,152,155)

#PCA_and_dendrograms.R

library(ape)
library(cluster)
library(bioDist)
source("/Users/npetraco/codes/R/chemometric_utilities/sourceme.R")

#Load spread sheet of ratios:
#File name components:
rootd<-"/Users/npetraco/latex/papers/graham_gas/data/"
raw<-read.csv("/Users/npetraco/latex/papers/graham_gas/data/BigMPD.csv",header=TRUE)

#Name labels for each type of MPD:
lbl<-raw[,1]
lbl.real<-as.character(raw[,2])
raw2<-raw[,3:ncol(raw)]
colnames(raw2)
raw3<-clean.up.spreadsheet(raw2)
X<-raw3

#Compute PCs
pca.model<-prcomp(X,center=TRUE,scale=TRUE)
summary(pca.model)
#Plot scree plot of PC variances:
plot(pca.model)

#Do a 2D PCA "scores" plot:
M<-2                                #Pick dimension
Z<-predict(pca.model)[,1:M]                 #Grab PCA scores
plot(Z[,1],Z[,2],col=lbl,pch=16,xlab="PC1",ylab="PC2",main="PC-scores") #Plot
text(Z[,1],Z[,2],labels=lbl,font=2,adj=1.5)       #Group lables
text(Z[,1],Z[,2],labels=1:nrow(X),font=1,adj=0)   #Obs. lables

#Do a 3D PCA "scores" plot:
M<-3                                #Pick dimension
Z<-predict(pca.model)[,1:M]                 #Grab PCA scores
open3d()                             #For RGL
plot3d(Z[,1],Z[,2],Z[,3],type="s",radius=0.1,col=as.numeric(lbl),aspect="iso",xlab="PC1",ylab=
"PC2",zlab="PC3")
text3d(Z[,1],Z[,2],Z[,3],text=lbl,font=1,adj=1.5) #Group lables
#snapshot3d("pls3d.png",fmt="png")

#Look at numerical values of PC variances:
summary(pca.model)
```

```
#Dendrogram of MPD relationships.
dmat<-dist(X) #Euclidian distance used.
dend<-hclust(dmat)
plot(dend,labels=lbl,col=lbl)

plot(as.phylo(dend),type="fan")

#Try other distances:
dmat<-man(X)
dmat<-cor.dist(X)
dmat<-KLD.matrix(X)
dend<-hclust(dmat)
plot(dend,labels=lbl,col=lbl)
```

```
#PCA_SVM_HooCV_Error-raes.R

library(ape)
library(cluster)
library(bioDist)
library(e1071)
library(caret)
source("/Users/npetraco/codes/R/chemometric_utilities/sourceme.R")

#Load spread sheet of ratios:
#File name components:
rootd<-"/Users/npetraco/latex/papers/graham_gas/data/"
raw<-read.csv("/Users/npetraco/latex/papers/graham_gas/data/BigMPD.csv",header=TRUE)

#Specific labels for each type of MPD.
#a1: original groupings
#a2: hypothesised related samples
lbl<-raw[,1]
lbl.real<-as.character(raw[,2])
raw2<-raw[,3:ncol(raw)]
colnames(raw2)
raw3<-clean.up.spreadsheet(raw2)
X<-raw3

a1<-
c(1,2,2,3,4,4,5,5,6,6,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,8,8,8,8,9,9,10,11,12,12,12,13,14,15,11,11,1
1,11,7,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,16,17,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,18,18,18,19,7,7,7,20,2,2,2,21,2,2,2,2,2,22,22
,23,24,24,24,25,26,27,28,29,29,30,31,32,33,33,34,34,35,36,36,36,37,38)
a2<-
c(43,7,46,98,57,76,116,161,87,54,146,159,140,110,79,9,78,103,86,53,36,26,37,38,39,67,64,102,
31,68,118,27,12,107,108,20,21,22,56,25,132,61,13,14,15,75,155,156,131,55,74,84,152,59,151,1
6,115,139,148,158,50,138,122,105,120,129,85,65,32,33,34,35,149,135,94,114,124,125,136,141,
6,60,96,88,52,51,71,83,92,63,90,89,66,133,144,154,160,8,47,80,93,104,117,119,147,106,123,13
0,134,145,23,137,17,49,70,81,91,58,162,97,5,150,112,24,101,10,28,72,163,29,48,73,42,18,143,1
57,109,3,126,127,44,1,2,4,95,82,11,113,62,153,41,121,100,77,19,142,111,128,40,69,45,30,99)
idxs<-NULL
for(i in 1:length(a2))
 {
  idx.map<-c(a2[i],a1[i])
  idxs<-rbind(idxs,idx.map)
 }
lbl2<-NULL
for(i in 1:nrow(idxs))
 {
  lbl.elm<-idxs[which(idxs[,1]==i),2]
  lbl2<-c(lbl2,lbl.elm)
 }
length(lbl2)
```

```
lbl3<-NULL
for(i in 1:length(lbl2))
 {
  lbl3<-c(lbl3,rep(lbl2[i],3) )
 }
length(lbl3)

#Compute PCs
pca.model<-prcomp(X,center=TRUE,scale=TRUE)
summary(pca.model)
#Plot scree of PC variances:
plot(pca.model)

#Do a 2D PCA "scores" plot:
M<-2                                #Pick dimension
Z<-predict(pca.model)[,1:M]                 #Grab PCA scores
plot(Z[,1],Z[,2],col=lbl3,pch=16,xlab="PC1",ylab="PC2",main="PC-scores") #Plot
text(Z[,1],Z[,2],labels=lbl3,font=2,adj=1.5)       #Group lables
text(Z[,1],Z[,2],labels=1:nrow(X),font=1,adj=0)   #Obs. lables

#Do a 3D PCA "scores" plot:
M<-3                                #Pick dimension
Z<-predict(pca.model)[,1:M]                 #Grab PCA scores
open3d()                             #For RGL
plot3d(Z[,1],Z[,2],Z[,3],type="s",radius=0.1,col=as.numeric(lbl3),aspect="iso",xlab="PC1",ylab
="PC2",zlab="PC3")
text3d(Z[,1],Z[,2],Z[,3],text=lbl3,font=1,adj=1.5) #Group lables
#snapshot3d("pls3d.png",fmt="png")

#Find optimal dimension with HOO-CV
err.vec<-NULL
ind.mat<-NULL
Mmax<-6
lbl<-lbl3
for(ii in 2:Mmax)
{
Z<-predict(pca.model)[,1:ii]
ind.vec<-NULL
for(i in 1:nrow(Z))
 {
  Z.heldout<-t(as.matrix(Z[i,]))
  lbl.heldout<-lbl[i]

  Z.kept<-Z[-i,]
  lbl.kept<-lbl[-i]
  svm.model<-svm(Z.kept,lbl.kept,scale=FALSE,type="C-
classification",kernel="linear",cost=0.1,fitted=TRUE,probability=TRUE)
  pred<-predict(svm.model,Z.heldout)
  #print(pred==lbl.heldout)
#  prob.vec<-attr(pred, "probabilities")[,]
```

```
  ind.vec<-c(ind.vec,pred==lbl.heldout)
  #print(hist(rowSums(ind.mat)))
 }
ind.mat<-cbind(ind.mat,ind.vec)
ccp<-(sum(ind.vec)/nrow(Z) )
err<-(1-ccp)*100
print(paste(ii,err))
err.vec<-c(err.vec,err)

}
cbind(2:Mmax,err.vec)
plot(2:Mmax,err.vec,typ="l")
min(err.vec)
which(err.vec==min(err.vec))
```

```
# CVA_and _Distance-metrics_with_ROC.R

library(ape)
library(cluster)
library(bioDist)
library(e1071)
library(caret)
library(MASS)
library(lattice)
library(ROCR)
library(bioDist)
source("/Users/npetraco/codes/R/chemometric_utilities/sourceme.R")
source("/Users/npetraco/latex/class/crj_80900/cross_validation_utilities.R")


#Load spread sheet of ratios:
#File name components:
rootd<-"/Users/npetraco/latex/papers/graham_gas/data/"
raw<-read.csv("/Users/npetraco/latex/papers/graham_gas/data/BigMPD.csv",header=TRUE)

#Name labels for each type of MPD:
lbl<-raw[,1]
lbl.real<-as.character(raw[,2])
raw2<-raw[,3:ncol(raw)]
colnames(raw2)
raw3<-clean.up.spreadsheet(raw2)
X<-raw3
dim(X)

#Compute PCs
pca.model<-prcomp(X,center=TRUE,scale=TRUE)
summary(pca.model)
#Plot histogram of PC variances:
plot(pca.model)

Mpc<-24                                 #Pick dimension
Zpc<-predict(pca.model)[,1:Mpc]                 #Grab PCA scores
#pairs(Zpc,col=lbl)

#Do CVA
lda.model<-lda(X,lbl)

#Canonical Variate "loadings". Called LDs:
Mcv<-3
Acv<-lda.model$scaling[,1:Mcv]
#Compute CVA "scores":
Zcv<-X %*% Acv
dim(lda.model$scaling[,])

#Make a 3D CVA "scores" plot:
```

```
open3d()                                #For RGL
plot3d(Zcv[,1],Zcv[,2],Zcv[,3],type="s",radius=0.9,col=as.numeric(lbl),aspect="iso",xlab="LD1
",ylab="LD2",zlab="LD3")
text3d(Zcv[,1],Zcv[,2],Zcv[,3],text=lbl,font=1,adj=1.5) #Group lables

#Canonical Variate "loadings". Called LDs:
Mcv<-3
Acv<-lda.model$scaling[,1:Mcv]
#Compute CVA "scores":
Zcv<-Zpc %*% Acv

#CVA HOO-CV error rate for choosen dimension:
#Note M <or= smaller of k-1 and p
hoocv.lda(X,lbl,M=33)
#~17% error with on CVA hoo-cv, grouping the samples by ID number.

Xs<-scale(X,center=TRUE,scale=TRUE)[,]
score.info<-correlation.compare3(Xs, lbl, lbl.real, cor.typ="spearman",
sames.warning.cutoff=0.5, difs.warning.cutoff=0.501, outfil.path=NULL)
score.info<-dist.compare(Xs, lbl, lbl.real, dist.typ="MI", sames.warning.cutoff=NULL,
difs.warning.cutoff=NULL, outfil.path=NULL)


#Score Histograms
hist(score.info[[1]],freq=FALSE,ylim=c(0,0.5),xlim=c(-2,30),col=rgb(0,1,0,1/4))
lines(density(score.info[[1]],width=0.15))
par(new=TRUE)
hist(score.info[[2]],freq=FALSE,ylim=c(0,0.5),xlim=c(-2,30),col=rgb(1,0,0,1/4))
lines(density(score.info[[2]],width=0.5))

#Stack columns to make box-whiskers plots
same.lbl<-rep("same",length(score.info[[1]]))
diff.lbl<-rep("different",length(score.info[[2]]))
sd.lbl<-factor(c(same.lbl,diff.lbl))
all.scores<-c(score.info[[1]],score.info[[2]])
score.tab<-data.frame(all.scores,sd.lbl)
bwplot(all.scores~sd.lbl,data=score.tab)

#ROC curve
pred<-prediction(all.scores,sd.lbl)
                    #y     #x
perf<-performance(pred,"tpr","fpr")
tpr.vals<-perf@y.values[[1]]
fpr.vals<-perf@x.values[[1]]
cutoffs<-perf@alpha.values[[1]]

roc.zoom<-cbind(cutoffs,100*tpr.vals,100*fpr.vals)
colnames(roc.zoom)<-c("cutoffs","TPR","FPR")
#roc.zoom
low<-0
```

```
high<-5.1
roc.zoom.plot<-roc.zoom[(roc.zoom[,3]<=high & roc.zoom[,3]>=low),]
plot(perf)
plot(roc.zoom.plot[,3],roc.zoom.plot[,2],typ="l",main="ROC Curve",xlab="False Positive Rate
(%)",ylab="True Positive Rate (%)")

#pearson 0.5312930
#tpr: 91.41104 (fnr: 8.58896)
#fpr: 5.015695

#kendall 0.5378788
#tpr: 96.72802
#fpr: 5.048516

#spearman: 0.6383690
#tpr: 95.09202
#fpr: 5.001389

#Examine PDR information metric:
pdr.dists(Xs,as.factor(lbl))
pdr.dist(Xs,lbl,152,155)

#Try a few other metrics.
#Closer the distance, the more similar the vectors:
cor.dist(Xs[1:2,])
1-cor(Xs[1,],Xs[2,])
KLD.matrix(Xs[1:2,])
KLdist.matrix(Xs[c(1,2),])
mutualInfo(Xs[c(1,4),])
#
x <- matrix(rnorm(100), nrow = 5)
KLD.matrix(x, method = "locfit", supp = range(x))
```

Subroutines used by R-scripts

```
# sourceme.R

# loads all subroutines from library – rootd.source should match local

library(rgl)

options(max.print=100000)

rootd.source<-"/Users/npetraco/codes/R/"

source(paste(rootd.source,"chemometric_utilities/cross_validation_utilities.R",sep=""))
source(paste(rootd.source,"chemometric_utilities/myImagePlot.R",sep=""))
source(paste(rootd.source,"chemometric_utilities/roc_utilities.R",sep=""))
source(paste(rootd.source,"chemometric_utilities/simca_utilities.R",sep=""))
source(paste(rootd.source,"chemometric_utilities/utilities.R",sep=""))
source(paste(rootd.source,"chemometric_utilities/variable.select.R",sep=""))
source(paste(rootd.source,"chemometric_utilities/score.compare.r",sep=""))
source(paste(rootd.source,"chemometric_utilities/pdr.dist.r",sep=""))
source(paste(rootd.source,"chemometric_utilities/normalize.r",sep=""))
```

```
# cross_validation_utilities.R

library(ape)
library(cluster)
library(bioDist)
library(e1071)
library(caret)
library(MASS)
library(lattice)
library(ROCR)
library(bioDist)
source("/Users/npetraco/codes/R/chemometric_utilities/sourceme.R")
source("/Users/npetraco/latex/class/crj_80900/cross_validation_utilities.R")


#Load spread sheet of ratios:
#File name components:
rootd<-"/Users/npetraco/latex/papers/graham_gas/data/"
raw<-read.csv("/Users/npetraco/latex/papers/graham_gas/data/BigMPD.csv",header=TRUE)

#Name labels for each type of MPD:
lbl<-raw[,1]
lbl.real<-as.character(raw[,2])
raw2<-raw[,3:ncol(raw)]
colnames(raw2)
raw3<-clean.up.spreadsheet(raw2)
X<-raw3
dim(X)

#Compute PCs
pca.model<-prcomp(X,center=TRUE,scale=TRUE)
summary(pca.model)
#Plot histogram of PC variances:
plot(pca.model)

Mpc<-24                                 #Pick dimension
Zpc<-predict(pca.model)[,1:Mpc]                 #Grab PCA scores
#pairs(Zpc,col=lbl)

#Do CVA
lda.model<-lda(X,lbl)

#Canonical Variate "loadings". Called LDs:
Mcv<-3
Acv<-lda.model$scaling[,1:Mcv]
#Compute CVA "scores":
Zcv<-X %*% Acv
dim(lda.model$scaling[,])

#Make a 3D CVA "scores" plot:
```

```
open3d()                                #For RGL
plot3d(Zcv[,1],Zcv[,2],Zcv[,3],type="s",radius=0.9,col=as.numeric(lbl),aspect="iso",xlab="LD1
",ylab="LD2",zlab="LD3")
text3d(Zcv[,1],Zcv[,2],Zcv[,3],text=lbl,font=1,adj=1.5) #Group lables

#Canonical Variate "loadings". Called LDs:
Mcv<-3
Acv<-lda.model$scaling[,1:Mcv]
#Compute CVA "scores":
Zcv<-Zpc %*% Acv

#CVA HOO-CV error rate for choosen dimension:
#Note M <or= smaller of k-1 and p
hoocv.lda(X,lbl,M=33)
#~17% error with on CVA hoo-cv, grouping the samples by ID number.

Xs<-scale(X,center=TRUE,scale=TRUE)[,]
score.info<-correlation.compare3(Xs, lbl, lbl.real, cor.typ="spearman",
sames.warning.cutoff=0.5, difs.warning.cutoff=0.501, outfil.path=NULL)
score.info<-dist.compare(Xs, lbl, lbl.real, dist.typ="MI", sames.warning.cutoff=NULL,
difs.warning.cutoff=NULL, outfil.path=NULL)


#Score Histograms
hist(score.info[[1]],freq=FALSE,ylim=c(0,0.5),xlim=c(-2,30),col=rgb(0,1,0,1/4))
lines(density(score.info[[1]],width=0.15))
par(new=TRUE)
hist(score.info[[2]],freq=FALSE,ylim=c(0,0.5),xlim=c(-2,30),col=rgb(1,0,0,1/4))
lines(density(score.info[[2]],width=0.5))

#Stack columns to make box-whiskers plots
same.lbl<-rep("same",length(score.info[[1]]))
diff.lbl<-rep("different",length(score.info[[2]]))
sd.lbl<-factor(c(same.lbl,diff.lbl))
all.scores<-c(score.info[[1]],score.info[[2]])
score.tab<-data.frame(all.scores,sd.lbl)
bwplot(all.scores~sd.lbl,data=score.tab)

#ROC curve
pred<-prediction(all.scores,sd.lbl)
                    #y     #x
perf<-performance(pred,"tpr","fpr")
tpr.vals<-perf@y.values[[1]]
fpr.vals<-perf@x.values[[1]]
cutoffs<-perf@alpha.values[[1]]

roc.zoom<-cbind(cutoffs,100*tpr.vals,100*fpr.vals)
colnames(roc.zoom)<-c("cutoffs","TPR","FPR")
#roc.zoom
low<-0
```

```
high<-5.1
roc.zoom.plot<-roc.zoom[(roc.zoom[,3]<=high & roc.zoom[,3]>=low),]
plot(perf)
plot(roc.zoom.plot[,3],roc.zoom.plot[,2],typ="l",main="ROC Curve",xlab="False Positive Rate
(%)",ylab="True Positive Rate (%)")

#pearson 0.5312930
#tpr: 91.41104 (fnr: 8.58896)
#fpr: 5.015695

#kendall 0.5378788
#tpr: 96.72802
#fpr: 5.048516

#spearman: 0.6383690
#tpr: 95.09202
#fpr: 5.001389

#Examine PDR information metric:
pdr.dists(Xs,as.factor(lbl))
pdr.dist(Xs,lbl,152,155)

#Try a few other metrics.
#Closer the distance, the more similar the vectors:
cor.dist(Xs[1:2,])
1-cor(Xs[1,],Xs[2,])
KLD.matrix(Xs[1:2,])
KLdist.matrix(Xs[c(1,2),])
mutualInfo(Xs[c(1,4),])
#
x <- matrix(rnorm(100), nrow = 5)
KLD.matrix(x, method = "locfit", supp = range(x))
```

```
#  myimagePlot.R
#This is code from: www.phaget4.org/R/myImagePlot.R

# ----- Define a function for plotting a matrix ----- #
myImagePlot <- function(x, ...){
     min <- min(x)
     max <- max(x)
     yLabels <- rownames(x)
     xLabels <- colnames(x)
     title <-c()
  # check for additional function arguments
  if( length(list(...)) ){
    Lst <- list(...)
    if( !is.null(Lst$zlim) ){
       min <- Lst$zlim[1]
       max <- Lst$zlim[2]
    }
    if( !is.null(Lst$yLabels) ){
       yLabels <- c(Lst$yLabels)
    }
    if( !is.null(Lst$xLabels) ){
       xLabels <- c(Lst$xLabels)
    }
    if( !is.null(Lst$title) ){
       title <- Lst$title
    }
  }
# check for null values
if( is.null(xLabels) ){
   xLabels <- c(1:ncol(x))
}
if( is.null(yLabels) ){
   yLabels <- c(1:nrow(x))
}

layout(matrix(data=c(1,2), nrow=1, ncol=2), widths=c(4,1), heights=c(1,1))

 # Red and green range from 0 to 1 while Blue ranges from 1 to 0
 ColorRamp <- rgb( seq(0,1,length=256),  # Red
                   seq(0,1,length=256),  # Green
                   seq(1,0,length=256))  # Blue
 ColorLevels <- seq(min, max, length=length(ColorRamp))

 # Reverse Y axis
 reverse <- nrow(x) : 1
 yLabels <- yLabels[reverse]
 x <- x[reverse,]

 # Data Map
 par(mar = c(3,5,2.5,2))
```

```
 image(1:length(xLabels), 1:length(yLabels), t(x), col=ColorRamp, xlab="",
 ylab="", axes=FALSE, zlim=c(min,max))
 if( !is.null(title) ){
   title(main=title)
 }
axis(BELOW<-1, at=1:length(xLabels), labels=xLabels, cex.axis=0.7)
 axis(LEFT <-2, at=1:length(yLabels), labels=yLabels, las= HORIZONTAL<-1,
 cex.axis=0.7)

 # Color Scale
 par(mar = c(3,2.5,2.5,2))
 image(1, ColorLevels,
     matrix(data=ColorLevels, ncol=length(ColorLevels),nrow=1),
     col=ColorRamp,
     xlab="",ylab="",
     xaxt="n")

 layout(1)
}
# ----- END plot function ----- #
```

```
# normalize.R
#######################################################################
#Normalizea spectrum (profile)
#IE-sets response to scale between 0 and 1
#######################################################################
norm01<-function(profile)
{

numNAs<-length(profile)-length(na.omit(profile))
nprofl<-na.omit(profile)
minp<-min(nprofl)
maxp<-max(nprofl)
nprofl<-apply(as.array(nprofl),1,function(x){(x-minp)/(maxp-minp)})
nprofl<-c(nprofl,rep(NA,numNAs))
return(nprofl)

}

#---------------------------------------------------------------
#L2 normalize a vector
#---------------------------------------------------------------
norml2<-function(signal)
{
 const<-1/sqrt(sum(signal*signal))
 normalized.signal <- (signal * const)
 return(normalized.signal)
}

#---------------------------------------------------------------
#Norm all the spectra (profiles) in a data matrix
#---------------------------------------------------------------
norm.stack<-function(profiles,typ)
{

num.profiles<-dim(profiles)[1]

if(typ=="norm01")
 {
  normsurf<-t(apply(profiles,1,norm01))
 }

if(typ=="norml2")
 {
  normsurf<-t(apply(profiles,1,norml2))
 }

return(normsurf)

}
```

```
# pdr.R

#-------------------------------------------------
#"Projected Difference Resolution" distances between
#GROUPS proposed by Harrington et al. to gauge
#pairwise group geparations
#-------------------------------------------------
pdr.dists<-function(dmat, lbls)
{

k<-nlevels(lbls)
pwi<-t(combn(k,2)) #pair-wise indices for group-to-group comparisons
#print(pwi)

all.pdr.dists<-rep(NA,nrow(pwi)) #initialize a vec for hold dists
for(i in 1:nrow(pwi))
 {
  idxa<-pwi[i,1]
  idxb<-pwi[i,2]
  #print(paste(idxa,idxb))

  a.list<-pick.out.groups(dmat,lbls,c(idxa))
  b.list<-pick.out.groups(dmat,lbls,c(idxb))
  dmat.a<-a.list[[1]]
  dmat.b<-b.list[[1]]
  a.avg<-colMeans(dmat.a)
  b.avg<-colMeans(dmat.b)
  avg.dif<-(a.avg-b.avg)

  ta<-dmat.a %*% avg.dif
  tb<-dmat.b %*% avg.dif

  Rab<-abs(mean(ta)-mean(tb))/(2*(sd(ta) + sd(tb)) )
  #print(Rab)
  all.pdr.dists[i]<-Rab

 }
#print(all.pdr.dists)
info<-cbind(pwi,all.pdr.dists)
colnames(info)<-c("Grp A","Grp B", "PDR dist")
return(info)

}

#-------------------------------------------------
#"Projected Difference Resolution" distance between
#two GROUPS proposed by Harrington et al.
#-------------------------------------------------
```

```
pdr.dist<-function(dmat, lbls, grpA, grpB)
{

a.list<-pick.out.groups(dmat,lbls,c(grpA))
b.list<-pick.out.groups(dmat,lbls,c(grpB))
dmatA<-a.list[[1]]
dmatB<-b.list[[1]]
a.avg<-colMeans(dmatA)
b.avg<-colMeans(dmatB)
avg.dif<-(a.avg-b.avg)

ta<-dmatA %*% avg.dif
tb<-dmatB %*% avg.dif

Rab<-abs(mean(ta)-mean(tb))/(2*(sd(ta) + sd(tb)) )

return(Rab)

}
```

```
# roc_utilities.R

#########################################################
#Pick two groups of data for ROC analysis
#For 1 vs. 1 just pick two groups
#For 1 vs. the rest, pick a group and say grp2="rest"
#NOTE: grp1 and grp2 can be names or numbers.
#########################################################
dichotomize.data<-function(dmat,alllbls,grp1,grp2)
{

#Grab group 1 indices
grp1.idx<-which(alllbls==grp1)

#generate label vector
lbl1<-rep(1,length(grp1.idx))

#Grab the chunk of group data out of dmat
xp1<-dmat[grp1.idx,]

#For 1 vs. rest:
if(grp2=="rest")
 {
  #Grab the rest of the data as group 2
  grp2.idx<-which(alllbls!=grp1)

  #generate label vector
  lbl2<-rep(2,length(grp2.idx))

  #Grab the chunk of group data out of dmat
  xp2<-dmat[grp2.idx,]

  #Reassemble into a new data matrix with lbl vector
  lblp<-factor(c(lbl1,lbl2))
  xnew<-rbind(xp1,xp2)
  dfnew<-cbind(lblp,xnew)
  dfnew<-data.frame(dfnew)

  #Spit out:
  return(dfnew)
 }

#Grab group 2 indices for 1 vs. 1
grp2.idx<-which(alllbls==grp2)

#generate label vector
lbl2<-rep(2,length(grp2.idx))

#Grab the chunk of group data out of dmat
xp2<-dmat[grp2.idx,]
```

```
#Assemble into a new data matrix with lbl vector
lblp<-factor(c(lbl1,lbl2))
xnew<-rbind(xp1,xp2)
dfnew<-cbind(lblp,xnew)
dfnew<-data.frame(dfnew)

#Spit out:
return(dfnew)

}

#########################################################
#Split data onto two Groups
#########################################################
split.data<-function(dmat,alllbls,grp1,grp2)
{

#Grab group 1 indices
grp1.idx<-NULL
for(i in 1:length(grp1))
 {
  grp1.idx<-c(grp1.idx,which(alllbls==grp1[i]))
 }

#generate label vector
lbl1<-rep(1,length(grp1.idx))

#Grab the chunk of group data out of dmat
xp1<-dmat[grp1.idx,]

#Grab group 2 indices
grp2.idx<-NULL
for(i in 1:length(grp2))
 {
  grp2.idx<-c(grp2.idx,which(alllbls==grp2[i]))
 }

#generate label vector
lbl2<-rep(2,length(grp2.idx))

#Grab the chunk of group data out of dmat
xp2<-dmat[grp2.idx,]

#Assemble into a new data matrix with lbl vector
lblp<-factor(c(lbl1,lbl2))
xnew<-rbind(xp1,xp2)
names(xnew)<-names(dmat)
dfnew<-cbind(lblp,xnew)
#dfnew<-data.frame(dfnew)
```

```
#Spit out:
#print(names(dmat))
#print(names(dfnew))
return(dfnew)

}
```

```
# scores.comparSAFE2.R

#-------------------------------------------------
#Compute correlation coefs "similarity" scores
#
#-------------------------------------------------
correlation.compare<-function(dat, lbls, actual.lbls, corr.warning.cutoff, outfil.path)
{

#num.grps<-nlevels(lbls)

#IMPORTANT: Assumes group labels are numerical, ordered and start from 1
grp.vec<-as.numeric(levels(lbls))

same.scores<-NULL
diff.scores<-NULL
mscgs<-NULL
for(i in 1:length(grp.vec))
  {
   #Label name for the test group:
   grp<-grp.vec[i]

   #Get the indices of the test group obs. vecs:
   #Use these for spitting out actual group labels later
   grp.idxs<-which(lbls==grp)

   #Labels names of the other groups:
   not.grp<-grp.vec[-i]

   #Get indices of other groups.
   #Unfortunately this is slow:
   not.grp.idxs<-which(lbls!=grp)
   not.lbl<-as.numeric(lbls[not.grp.idxs])
   #print(length(not.lbl))

   #Compute the correlation coefs between data vecs from the same group
   #Grab vecs from the same group
   tmp.same<-pick.out.groups(dat,lbls,c(grp))
   dat.same<-as.matrix(tmp.same[[1]])

   #Compute correlations between vecs.
   #Note, need more than 1 observation in the test group
   if(length(grp.idxs)>1)
    {
     tmp.same<-cor(t(dat.same))
     grp.grp.scores<-extract.lower.triangle(tmp.same)

     #Store the coefs
     same.scores<-c(same.scores,grp.grp.scores)
    }
```

```
  #Separate out data of other groups
  tmp.diff<-pick.out.groups(dat,lbls,not.grp)
  dat.diff<-as.matrix(tmp.diff[[1]])
  #print(nrow(dat.diff))
  #print("")

  for(j in 1:nrow(dat.same))
   {
    for(k in 1:nrow(dat.diff))
      {
       #print(paste(grp,"",not.lbl[k] ))
       if(grp < not.lbl[k])
        {
         #print(paste("GOOD!","Compare grp:",grp,"to grp:",not.lbl[k]))
         grp.notgrp.score<-cor(dat.same[j,],dat.diff[k,])
         if(grp.notgrp.score>corr.warning.cutoff)
           {
            mscgs<-rbind(mscgs,paste("UH
OH!:",actual.lbls[grp.idxs[j]],"vs.",actual.lbls[not.grp.idxs[k]],"CORR=",grp.notgrp.score))
           }
         diff.scores<-c(diff.scores,grp.notgrp.score)
         #print(grp.notgrp.score)
        }
#        if(grp > not.lbl[k])
#         {
#          print(paste("BAD!","Grp:",grp,"greater than grp:",not.lbl[k],"DON'T COMPARE
AGAIN"))
#         }
#        if(grp == not.lbl[k])
#         {
#          print(paste("**********************************REALLY BAD!","Grp:",grp,"is the
same as grp:",not.lbl[k],"IT'S NOT SUPPOSED TO BE!!!!!!"))
#         }

      }
    }

  print(paste("Done with group:",i))

 }

print(mscgs)
write.table(mscgs,file=paste(outfil.path,"Warning_Messages_",as.character(corr.warning.cutoff),
".txt",sep=""))
#print(paste(outfil.path,"Warning_Messages_",as.character(corr.warning.cutoff),".txt",sep=""))

return(list(same.scores,diff.scores))
```

```
}

#-------------------------------------------------
#Compute correlation coefs "similarity" scores but
#give more diagnostic info
#
#-------------------------------------------------
correlation.compare3<-function(dmat, lbls, actual.lbls, cor.typ, sames.warning.cutoff,
difs.warning.cutoff, outfil.path)
{

k<-nlevels(lbl) #num grps
n<-nrow(dmat)    #num obs
#Total number of comparisons:
total.num.comp<-choose(n,2)

#obs. indices of unique pairs of comparisons
comp.idxs<-combn(n,2) #note, conbinations are dumped out in column-wise format

info.mat<-matrix(rep("x",total.num.comp*6),nrow=total.num.comp,ncol=6) #initialize a
character mat to hold info
#info.mat<-matrix(rep("x",10*6),nrow=10,ncol=6)
print(paste(" Starting similarity score computations:",date()))
tim<-system.time(
for(i in 1:ncol(comp.idxs))
#for(i in 1:10)
{
 idxa<-comp.idxs[1,i]
 idxb<-comp.idxs[2,i]
 grpa<-lbls[idxa]
 grpb<-lbls[idxb]
 actual.grpa<-actual.lbls[idxa]
 actual.grpb<-actual.lbls[idxb]
 sscore<-cor(dmat[idxa,],dmat[idxb,],method=cor.typ) #compute the similarity scores here
 info.vec<-c(as.character(grpa),as.character(grpb),as.character(actual.grpa),
as.character(actual.grpb), grpa==grpb, sscore)
 info.mat[i,]<-info.vec
 #print(info.vec)
}
)
print(paste("Done with similarity score computations:",date()))
print(tim)
rownames(info.mat)<-NULL
colnames(info.mat)<-c("Grp a","Grp b","Name a","Name b", "Same GrpQ","Sim score")
#print(info.mat)

num.same.comp<-sum(as.logical(info.mat[,5]))
num.dif.comp<-total.num.comp-num.same.comp
```

```
same.idxs<-which(as.logical(info.mat[,5])==TRUE) #pull out same/different group comparison
scores
dif.idxs<-which(as.logical(info.mat[,5])==FALSE)

#Process comparisons between obs of the SAME group
same.mat<-info.mat[same.idxs,]
#print(same.mat)
same.scores<-as.numeric(same.mat[,6])

problem.idxs.same<-which(same.scores<=sames.warning.cutoff) #pull out the problem scores
problem.same.mat<-same.mat[problem.idxs.same,]
#print(problem.same.mat)

#Process comparisons between obs between DIFFERENT groups
dif.mat<-info.mat[dif.idxs,]
#print(dif.mat)
dif.scores<-as.numeric(dif.mat[,6])

problem.idxs.dif<-which(dif.scores>=difs.warning.cutoff) #pull out the problem scores
problem.dif.mat<-dif.mat[problem.idxs.dif,]
#print(nrow(problem.dif.mat))

print(paste("Total number of comparisons:           ",total.num.comp))
print(paste("Number of SAME group comparisons:      ",num.same.comp,",
",length(problem.idxs.same)/num.same.comp*100,"% a problem; below threshold:
",sames.warning.cutoff, sep=""))
print(paste("Number of DIFFERENT group comparisons: ",num.dif.comp,",
",length(problem.idxs.dif)/num.dif.comp*100,"% a problem; above threshold:
",difs.warning.cutoff, sep=""))

#Write scores to file for troubleshooting:
write.table(problem.dif.mat,file=paste(outfil.path,"Problem_Observations_Between_Different_G
roups",as.character(difs.warning.cutoff),".txt",sep=""))
write.table(problem.same.mat,file=paste(outfil.path,"Problem_Observations_Within_Same_Gro
ups",as.character(sames.warning.cutoff),".txt",sep=""))

write.table(dif.mat,file=paste(outfil.path,"All_Score_Info_Between_Different_Groups",".txt",sep
=""))
write.table(same.mat,file=paste(outfil.path,"All_Score_Info_Within_Same_Groups",".txt",sep="
"))

return(list(same.scores,dif.scores))
}
```

```
# simca_utilities.R

###################################
#Examine the PCA for each group:
###################################
group.pca.explore<-function(datmat,lbls,centerQ,scaleQ,printQ)
{

k<-nlevels(lbls)
lnames<-levels(lbls)

#Generate a PCA for each group
pca.model.list<-NULL
for(i in 1:k)
 {
  grp.idxs<-which(lbls==lnames[i])
  datmat.sub<-datmat[grp.idxs,]
  #print(dim(Xtr.sub))
  pca.model.grp<-prcomp(datmat.sub,center=centerQ,scale=scaleQ)
  #Tack results into running list:
  pca.model.list<-c(pca.model.list,list(pca.model.grp))
  #Print out model info if desired
  if(printQ==TRUE) print(summary(pca.model.grp))
 }

#Spit back list of group PCA models:
return(pca.model.list)

}


################################################
#With dimension choices, build each SIMCA model:
################################################
simca.models<-function(datmat,lbls,centerQ,scaleQ,dim.choices)
{

k<-nlevels(lbls)
lnames<-levels(lbls)

Z.list<-NULL
var.list<-NULL
od.list<-NULL
sd.list<-NULL
od.cut<-NULL
sd.cut<-NULL
for(i in 1:k)
 {
  grp.idxs<-which(lbls==lnames[i])
  datmat.sub<-datmat[grp.idxs,]
```

```
  #Inefficient but rerun PCA. Eventually change!

  #PCA for each model
  pca.model<-prcomp(datmat.sub,center=centerQ,scale=scaleQ) #Run PCA
  M<-dim.choices[i]                          #Pick dimension
  Z<-predict(pca.model)[,1:M]                    #Grab PC scores
  var.list<-c(var.list,list(pca.model$sdev[1:M]^2))  #Grab variances
  Z.list<-c(Z.list,list(Z))                  #Tack scores into list of models

  #Orthagonal distances for each model
  Apc<-pca.model$rotation[,1:M]                  #Grab PC loadings
  datmat.proj<-Z%*%t(Apc)                        #Project scores back up to data space
  #Prep datmat.sub if pca was centered or scaled:
  datmat.sub<-scale(datmat.sub,center=centerQ,scale=scaleQ)[,]

  res.mat<-(datmat.sub-datmat.proj)              #Compute difference between full data set and
PCA deduced model
  orthag.dists<-sqrt(rowSums(res.mat * res.mat))     #Orthagonal dists. of each obs to PCA model
  od.list<-c(od.list,list(orthag.dists))
  cutoff.od<-(median(orthag.dists^(2/3)) + mad(orthag.dists^(2/3))*qnorm(0.975))^(3/2)
  od.cut<-c(od.cut,cutoff.od)                    #Accumulate od cutoffs

  #Score distances for each model
  score.dists<-sqrt(mahalanobis(Z,colMeans(Z),cov(Z))) #Compute Mahalanobis "Score
distances"
#  score.dists<-sqrt(mahalanobis(Z,colMeans(Z),diag(pca.model$sdev^2))) #Compute
Mahalanobis "Score distances"
  sd.list<-c(sd.list,list(score.dists))
  cutoff.sd<-sqrt(qchisq(0.975,M))               #Huber cutoff for SD outliers
  sd.cut<-c(sd.cut,cutoff.sd)                    #Accumulate sd cutoffs
 }

info.list<-list(Z.list,var.list,od.list,sd.list,od.cut,sd.cut)
return(info.list)

}


#############################################################################
#Compute "Score Distance" of a PCA projected data vector from a PCA model
#############################################################################
score.distance<-function(Zi.unk,pca.vars)
{

tmp<-(Zi.unk * Zi.unk)
tmp2<-sum(tmp*(1/pca.vars))
scd<-sqrt(tmp2)
return(tmp2)

}
```

```
#############################################################################
######
#Compute "Orthagonal Distance" of a back PCA projected data vector from data vector
#############################################################################
######
orthagonal.distance<-function(Xi.unk,Xi.proj.unk)
{
 return(sqrt(sum((Xi.unk-Xi.proj.unk)^2)))
}


#########################
#Assign groups via simca
#########################
simca.classify<-function(datmat.te,lbls,dim.choices,pca.model.list,simca.models.list,tunep)
{

k<-nlevels(lbls)
lnames<-levels(lbls)
nobs<-dim(datmat.te)[1]
sd.cutoffs<-simca.models.list[[6]]
od.cutoffs<-simca.models.list[[5]]

ids<-NULL
for(i in 1:nobs) #loop over all unknown obs vects
 {
  tobs<-as.matrix(datmat.te[i,]) #Grab a obs vec to be classified. Class matrix needed.
  sds<-NULL
  ods<-NULL
  for(j in 1:k)       #Loop over each group. FIX! This may get slow.
   {
    grp.cent<-pca.model.list[[j]]$center
    grp.scal<-pca.model.list[[j]]$scale
#    print(class(grp.cent))
#    print(class(grp.scal))
#    print(dim(tobs))

    #Scale obs vect same way as the group:
    tobs.scaled<-scale(tobs,center=grp.cent,scale=grp.scal)[,]
#    print(tobs.scaled)

    #Project obs vect into group PCA model:
    grp.M<-dim.choices[j]
    grp.Apc<-pca.model.list[[j]]$rotation[,1:grp.M]
    ztobs <- tobs.scaled %*% grp.Apc
#    print(ztobs)

    #Compute score distance to group:
    grp.vars<-(pca.model.list[[j]]$sdev[1:grp.M])^2
```

```
    sd.unk<-score.distance(ztobs,grp.vars)
    sds<-c(sds,sd.unk)

    #Compute orthagonal distance to group:
    tobs.proj <- ztobs %*% t(grp.Apc)
#     print(tobs.proj)
    od.unk<-orthagonal.distance(tobs.scaled,tobs.proj)
    ods<-c(ods,od.unk)
   }

  sds<-((1-tunep) * (sds/sd.cutoffs))
  ods<-(tunep * (ods/od.cutoffs))
  simca.dists<-colSums(rbind(sds,ods))
#  print(simca.dists)
  id<-lnames[which(simca.dists==min(simca.dists))]
  ids<-c(ids,id)
 }

ids<-factor(ids)
return(ids)

}
```

```
# utilities.R

#---------------------------------------------------------------
#Make up a lable vector according to number of samples
#---------------------------------------------------------------
generate.label.vec<-function(num.samps.vec)
{

lbl.vec<-NULL
for(i in 1:length(num.samps.vec) )
 {
  grpids<-rep(i,num.samps.vec[i])
  lbl.vec<-factor(c(lbl.vec,grpids))
 }

return(lbl.vec)

}

#---------------------------------------------------------------
#Counts the number of replicates for each group.
#Should be independent of group naming convention
#---------------------------------------------------------------
count.group.replicates<-function(arb.lbls)
{

num.samps.vec<-sapply(as.numeric(levels(factor(arb.lbls))), function(x){sum(arb.lbls==x)})
return(num.samps.vec)

}

#---------------------------------------------------------------
#Pick out groups of observations and form a new X matrix
#---------------------------------------------------------------
pick.out.groups<-function(X.mat,all.lbls,grp.picks)
{

pick.out.rows<-NULL
new.grp.lbls<-NULL
for(i in 1:length(grp.picks))
 {
  grp.idxs<-which(as.numeric(all.lbls)==as.numeric(grp.picks[i]))
  pick.out.rows<-c(pick.out.rows,grp.idxs)
  #print(grp.idxs)
  new.grp.lbl<-rep(i,length(grp.idxs))
  new.grp.lbls<-c(new.grp.lbls,new.grp.lbl)
 }

new.grp.lbls<-factor(new.grp.lbls)
new.X.mat<-X.mat[pick.out.rows,]
```

```
return(list(new.X.mat,new.grp.lbls))
 }


#-----------------------------------------------------------------
#Extract the lower trianfle of elements in a square matrix
#-----------------------------------------------------------------
extract.lower.triangle<-function(sqmat)
{

lt.vec<-NULL
for(i in 1:dim(sqmat)[1])
  {
   for(j in 1:dim(sqmat)[2])
     {
      if(i<j)
       {
           lt.vec<-c(lt.vec,sqmat[i,j])
       }
     }
  }
return(lt.vec)
}


#-----------------------------------------------------------------
#Get empty value indicators out of a spread sheet and impute with 0s out
#-----------------------------------------------------------------
clean.up.spreadsheet<-function(raw.dmat)
{

X<-NULL
for(i in 1:dim(raw.dmat)[2])
 {
  #Grab a row from the spread sheet
  xcol<-raw.dmat[,i]
  #Pick out the erronious special characters and replace with 0:
  xcol<-
replace(as.character(xcol),c(which(xcol==""),which(xcol=="#VALUE!"),which(xcol=="#DIV/0
!")),0)
  #Replace NAs with 0:
  xcol[is.na(xcol)]<-0
    xcol<-as.numeric(xcol)

  #Replace erronious negative ratios with 0:
  xcol[which(xcol<0)]<-0

  #Tack cleaned up row into matrix:
  X<-cbind(X,xcol)
 }
```

```
rownames(X)<-NULL
colnames(X)<-NULL
#X<-matrix(as.numeric(X),nrow=nrow(X),ncol=ncol(X))
return(X)

}

#---------------------------------------------------
#Check angles between vectors.
#Useful for CVA
#---------------------------------------------------
arg<-function(vector1,vector2)
{
v1<-as.matrix(vector1,nrow=length(vector1))
v1n<-(v1/norm(v1,type="F"))

v2<-as.matrix(vector2,nrow=length(vector2))
v2n<-(v2/norm(v2,type="F"))
dp<-as.numeric(as.character(t(v1n)%*%v2n))
angle<-acos(dp) * 180/pi #angle units are degrees

return(angle)

}

#---------------------------------------------------
#Index to subscripts function for a retangular matrix
#---------------------------------------------------
ind2sub<-function(idx,num.row)
{
# rc.ind <- c(row(M)[ind], col(M)[ind] )
r = ((idx-1) %% num.row) + 1
c = floor((idx-1) / num.row) + 1
subscrpt<-c(r,c)

return(subscrpt)
}

#---------------------------------------------------
#Subscripts to index function for a retangular matrix
#---------------------------------------------------
sub2ind<-function(r,c,num.cols)
{

idx<-((r-1)*num.cols + c)
 return(idx)
}
```

```
# variable.select.R
library(rgl)

#-----------------------------------------------------------------
#Compute univariate Fisher ratios between two samples
#Reference: Sahota and Morgan, Anal.Chem.64,2383,1992
#-----------------------------------------------------------------
fisher.ratios<-function(dmat,lbls,grp1,grp2,plotQ=TRUE,plot.typ="l")
{

grp1.idxs<-which(lbls==grp1)
grp2.idxs<-which(lbls==grp2)

means.g1<-colMeans(dmat[grp1.idxs,])
means.g2<-colMeans(dmat[grp2.idxs,])

var.g1<-apply(dmat[grp1.idxs,],2,var)
var.g2<-apply(dmat[grp2.idxs,],2,var)
#var.g1<-var(dmat[grp1.idxs,])
#var.g2<-var(dmat[grp2.idxs,])

ratios<-((means.g1-means.g2)^2)/(var.g1+var.g2)

if(plotQ==TRUE)
 {
  plot(1:length(ratios),ratios,type=plot.typ)
 }

return(ratios)

}


#-----------------------------------------------------------------
#3D rotatable bar plot. Good for visualizing loadings across
#many variables.
#-----------------------------------------------------------------
loading.bar3D<-function(loading.mat,bar.width)
{

dr<-dim(loading.mat)[1]
dc<-dim(loading.mat)[2]

bp.mat<-NULL
for(i in 1:dr)
 {
  for(j in 1:dc)
   {
```

```
    bp.vec<-c(i,j,loading.mat[i,j])
    bp.mat<-rbind(bp.mat,bp.vec)
   }
 }
bp.mat<-abs(bp.mat)

#X-axis is loadings. Y-axis is variables.

yLabels <- c(1:nrow(loading.mat))
xLabels <- rownames(loading.mat)
print("NOTE: ABSOLUTE VALUES OF LOADINGS DISPLAYED!")

plot3d(bp.mat[,1],bp.mat[,2],bp.mat[,3],axes=T,type="h",lwd=bar.width,xlab="Variables",ylab=
"L.V. #",zlab="|Loading|")
axis3d(edge='x-', at=1:length(xLabels), labels=xLabels, cex.axis=0.7)
axis3d(edge='y-',at=1:length(yLabels), labels=yLabels, cex.axis=0.7)

}


# End of R scripts and subroutines
```