PNNL Chemometric Methods

Table of Contents

Classical Least Squares (CLS)

Classical least squares computes C_{cls} that minimizes $||CK_{\text{cls}} - A_{\text{unkown}}||_2$, where $CK = A$ is the Beer's law relationship between concentration C, extinction coefficient K, and absorbtion A. Multiplier matrix B_{cls} is the pseudo-inverse of Beer's law extinction coefficient matrix K_{cls} such that $C_{\text{cls}} = A_{\text{unknown}} \cdot B_{\text{cls}}$. The CLS algorithm is encapsulared in function `pnnl_cls`.

```
function [C_cls, B_cls, K_cls] = pnnl_cls(A_train, C_train, A_unknown)

    K_cls = C_train \ A_train;

    C_cls = A_unknown / K_cls;

    B_cls = pinv(K_cls);

end
```

Principal Component Regression (PCR)

Principal component regression computes C_{pcr} using C_{train} and the first r principal components of A_{train}. The PCR algorithm is encapsulated in function `pnnl_pcr`.

```
function [C_pcr, B_pcr] = pnnl_pcr(A_train, C_train, A_unknown, r)

    [U,S,V] = svd(A_train,'econ');

    B_pcr = V(:,1:r) / S(1:r,1:r) * U(:,1:r)' * C_train;

    C_pcr = A_unknown * B_pcr;

end
```

Partial Least Squares (PLS)

Partial least squares computes C_{pls} using the weights from the SIMPLS algorithm with r latent variables. The PLS algorithm without mean centering is encapsulated in function `pnnl_pls`.

```
function [C_pls, B_pls] = pnnl_pls(A_train, C_train, A_unknown, r)

    X = A_train;
    Y = C_train;

    [X_loadings, Y_loadings, X_scores, Y_scores, Weights] = pnnl_simpls(X, Y, r);

    B_pls = Weights * Y_loadings';

    C_pls = A_unknown * B_pls;

end
```

Napalm Data

Load the included napalm data to run the CLS, PCR, and PLS algorithms.

```
clearvars
```

```
load pnnl_napalm_data
whos
```

```
  Name                    Size              Bytes  Class     Attributes

  A_train                20x1713           274080  double
  A_unknown              12x1713           164448  double
  C_train                20x3                 480  double
  C_validation           12x3                 288  double
  ConcentrationUnits      1x4                   8  char
  ConstituentNames        1x3                 364  cell
  WavenumberLabel         1x20                 40  char
  Wavenumbers             1x1713            13704  double
```

For example, predict C_{pcr} with 3 principal components.

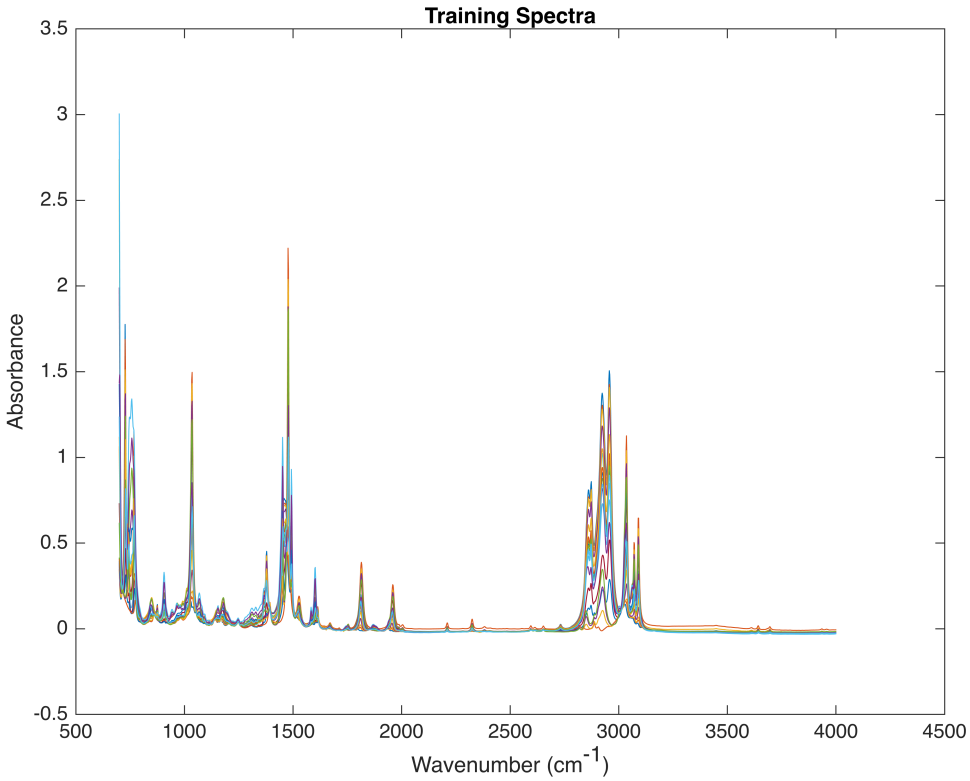
```
C_pcr = pnnl_pcr(A_train, C_train, A_unknown, 3)
```

```
C_pcr = 12×3
   15.7808   42.3993   44.0402
   14.2776   43.9051   44.7869
   16.0949   41.5656   43.6256
   22.8143   40.7572   40.2362
   23.0956   35.6576   42.6710
   17.5895   38.0799   47.4498
   16.8950   40.4319   45.4173
   17.4393   38.9973   46.2379
   23.0404   41.6075   38.9080
   22.2427   42.2424   37.7980
     ⋮
```

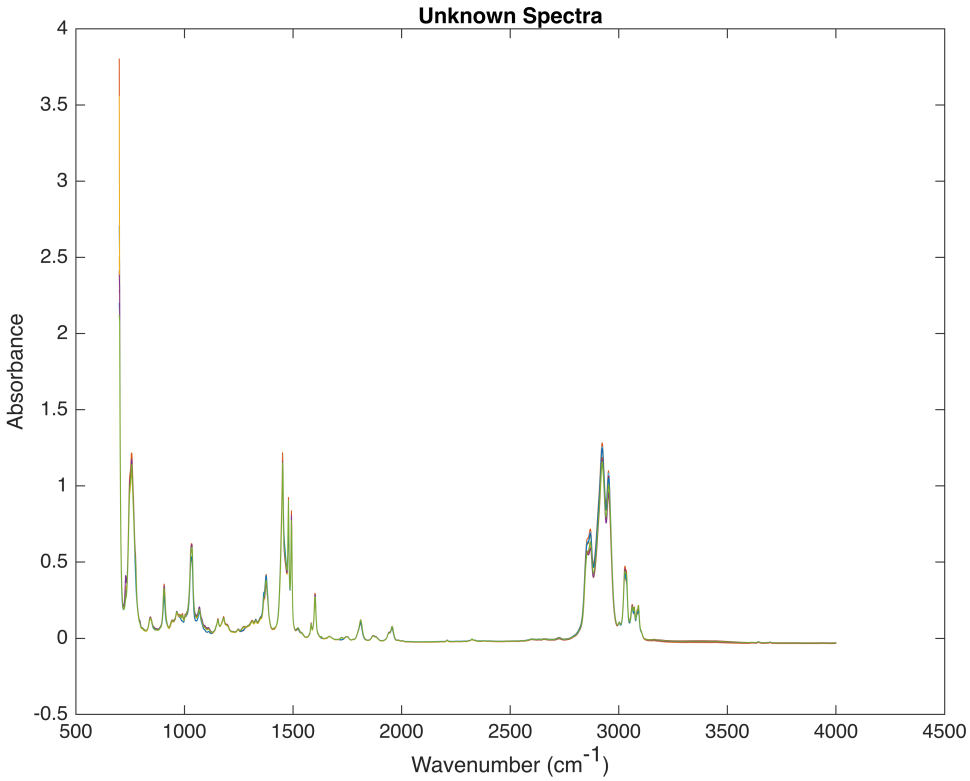
Examine the Data

It's a good idea to look at the data before analyzing it.

```
plot(Wavenumbers,A_train)
xlabel(WavenumberLabel)
ylabel('Absorbance')
title('Training Spectra')
```




```
plot(Wavenumbers,A_unknown)
xlabel(WavenumberLabel)
ylabel('Absorbance')
title('Unknown Spectra')
```



Root Mean Square Error (RMSE)

The root mean square error of the prediction C_{pcr} is $\text{RMSEP} = \sqrt{\text{mean}\left((C_{\text{pcr}} - C_{\text{validation}}).^2\right)}$. The RMSE algorithm is encapsulated in function `pnnl_rmse`. Using `vecnorm` is theoretically equivalent, but is better than computing the square and square root with respect to very large or very small numbers.

```
function rmse = pnnl_rmse(C_computed,C_actual)
    p = size(C_actual,1);
    rmse = vecnorm(C_computed - C_actual, 2, 1) / sqrt(p);
end
```

The columns correspond to the RMSEP of the compnents benzene, polystyrene, and gasolene.

```
RMSEP_pcr = pnnl_rmse(C_pcr, C_validation) %#ok<*NASGU>
```

```
RMSEP_pcr = 1×3
    5.1766    2.7318    6.2906
```

Calibration Predictions

Training

You can run a predictor method against the training data.

```
C_pcr_train = pnnl_pcr(A_train, C_train, A_train, 3);
RMSEC = pnnl_rmse(C_pcr_train, C_train)
```

```
RMSEC = 1×3
    4.6738    2.3109    5.9859
```

Cross-validation

You can cross-validate by running a predictor method against the training data. For each of the rows in the training set, leave one row out, and use that row as the unknown data. The `pnnl_cross_validation` function encapsulates this,

```
function C_cross_validation = pnnl_cross_validation(method, A, C, varargin)
    [p,n] = size(C);
    C_cross_validation = zeros(p,n);
    for i = 1:p
        C_cross_validation(i,:) = method(A([1:i-1,i+1:p],:),C([1:i-1,i+1:p],:),A(i,:),varargin{:});
    end
end
```

where method is a function handle,

```
method = @pnnl_cls
```

or

```
method = @pnnl_pcr
```

or

```
method = @pnnl_pls
```

and the variable input argument `varargin` is the number of principal components for PCR or the number of latent variables for PLS.

For example

```
C_pcr_cross_validation = pnnl_cross_validation(@pnnl_pcr, A_train, C_train, 3);
RMSECV = pnnl_rmse(C_pcr_cross_validation, C_train)
```

```
RMSECV = 1×3
    5.9206    2.8541    7.4667
```

Plot the concentration results

Plot the results of $C_{\text{validation}}$ against C_{pcr} using a scatter plot

```
figure
plot(C_validation,C_pcr,'.','MarkerSize',35);
```

Save the color order of the constituents to coordinate the colors of the training and cross-validation data.

```
colorOrder = get(gca,'ColorOrder');
```

Hold the axis and plot the training and cross-validation sets.

```
hold on
```

Plot training prediction with o marker.

```
plot(C_train,C_pcr_train,'o','MarkerSize',10,'LineWidth',1)
```

Plot cross-validation concentrations with + marker.

```
plot(C_train,C_pcr_cross_validation,'+','MarkerSize',10,'LineWidth',1)
```

Draw a 1-1 line.

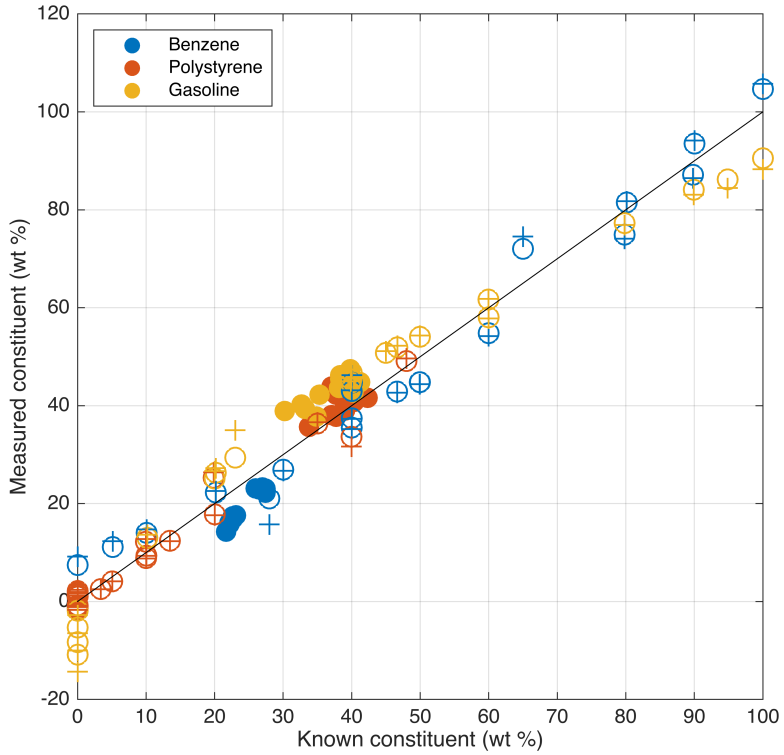
```
line(C_train,C_train,'Color','k');
```

Reset the color order so the training and cross-validation sets have the same colors for their constituents.

```
n = size(C_validation,2);
set(gca,'ColorOrder',colorOrder(1:n,:));
```

Create a legend and make the axis square, so you can see that the training line is 45 degrees.

```
legend(ConstituentNames{:},'Location','northwest')
xlabel('Known constituent (wt %)')
ylabel('Measured constituent (wt %)')
axis square
grid on
box on
```



Unknown prediction is displayed as a solid dot. Training predicition is displayed as an open circle. Cross-validation prediction is displayed as a plus sign.

Functions that load napalm data, predict, and plot

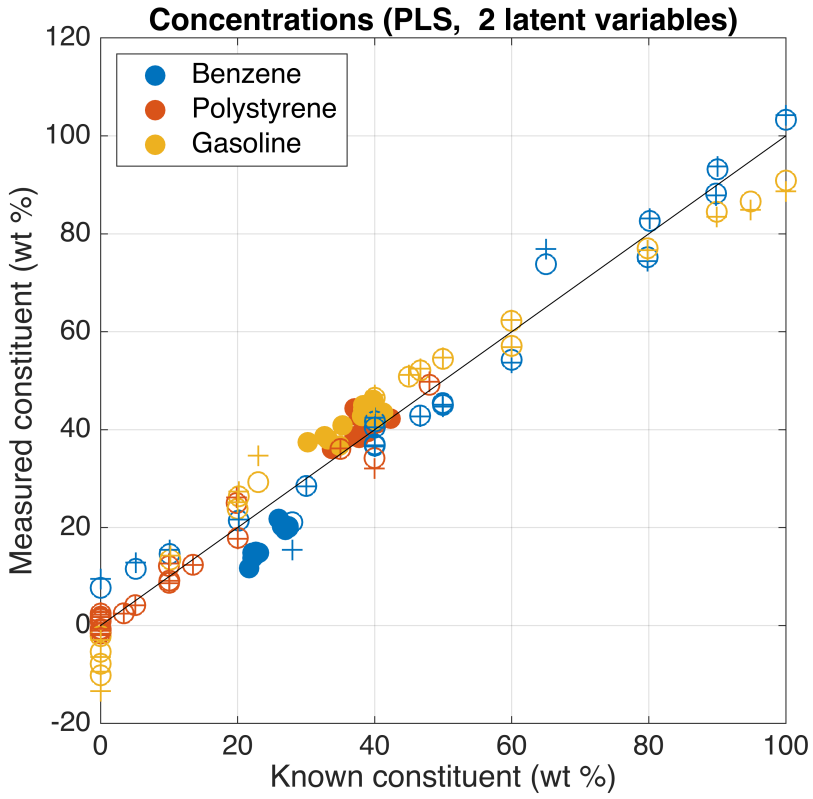
The following functions encapsulate the steps in the previous sections for each of the methods. They load the napalm data, predict and plot the concentration matrices, and display the RMSE values.

```
pnnl_napalm_cls
pnnl_napalm_pcr(nPrincipalComponents)
meanCentered = true;
pnnl_napalm_pls(nLatentVariables, meanCentered)
```

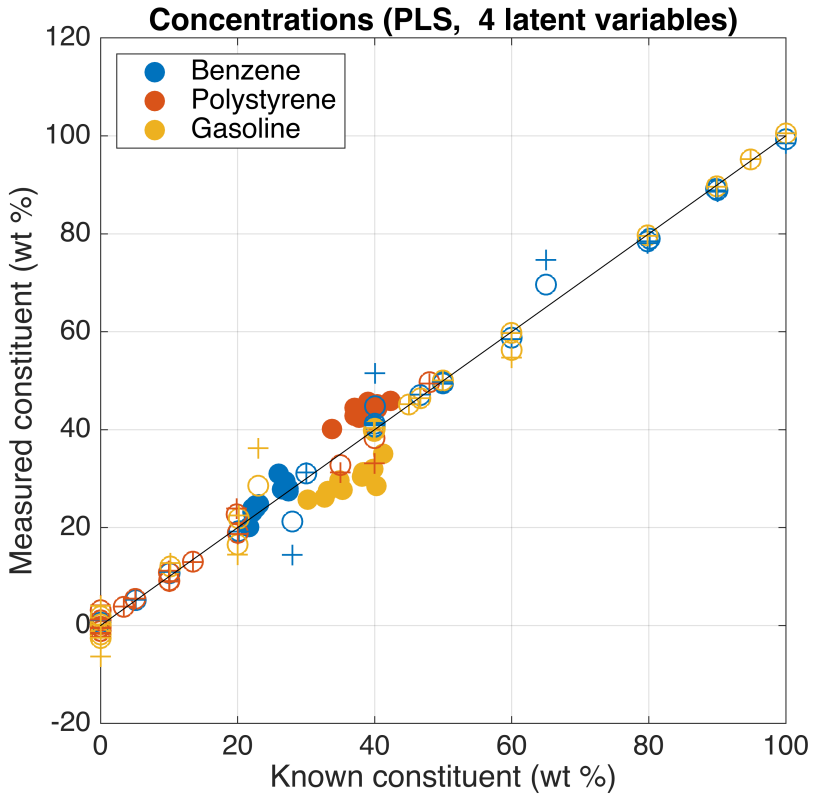
Plotting

If you call them with no output arguments, then the plots and displays are shown. If you input a vector to PCR and PLS, then they are run for the number of principal components or latent variables, respectively.

```
meanCentered = true;
pnnl_napalm_pls([2 4 8], meanCentered);
```

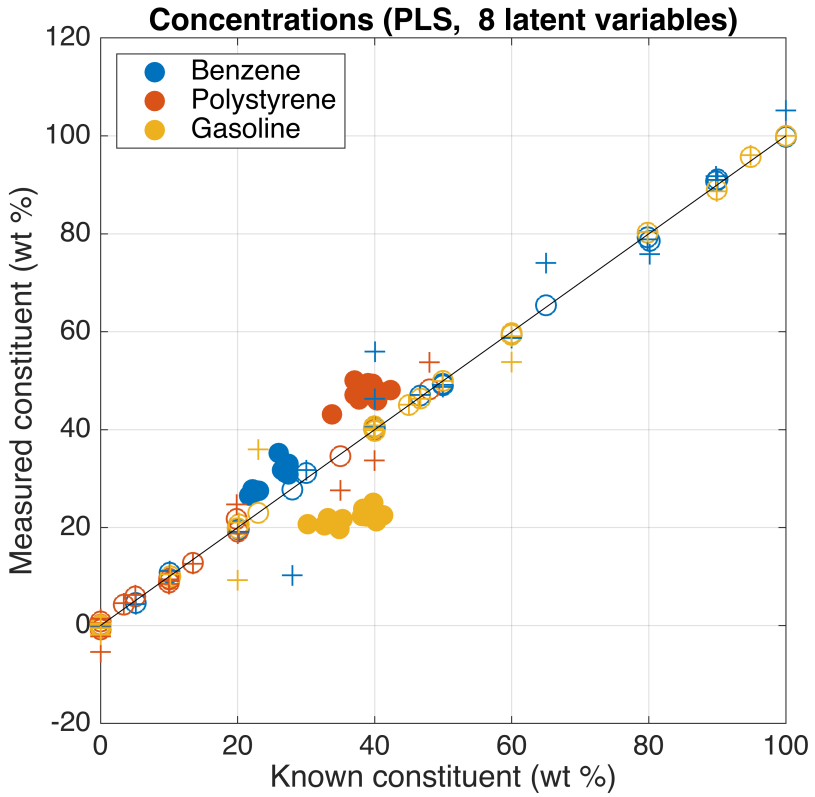



```
Legend: Dot is predicted. Circle is train. Cross is cross-validation.
------------------------------------------------------------------------
  PLS,  2 latent variables       Benzene    Polystyrene      Gasoline
                    RMSEC        4.5616         2.2064        5.8133
                   RMSECV        5.8788         2.7588        7.2286
                    RMSEP        7.5077           3.04        5.0846
------------------------------------------------------------------------
```



Legend: Dot is predicted. Circle is train. Cross is cross-validation.

PLS, 4 latent variables	Benzene	Polystyrene	Gasoline
RMSEC	2.2574	1.3926	1.9696
RMSECV	4.6479	2.4087	3.9235
RMSEP	2.0484	5.4414	7.344




```
Legend: Dot is predicted. Circle is train. Cross is cross-validation.
--------------------------------------------------------------------
  PLS,  8 latent variables       Benzene    Polystyrene      Gasoline
                    RMSEC        0.68442        0.71824       0.41195
                   RMSECV         6.1219         3.1259        4.0761
                    RMSEP         5.4028          9.227        14.919
--------------------------------------------------------------------
```

Output Data

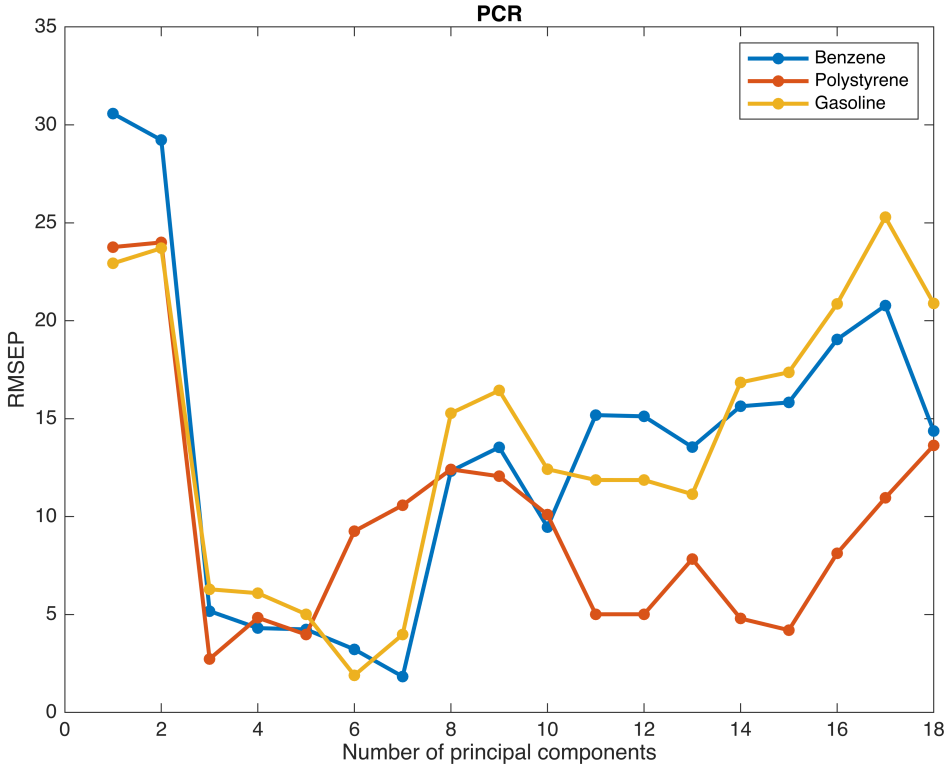
The computed data are returned if you call the functions with output arguments, and the plot and display are suppressed.

```
[C_cls, RMSEP_cls, C_cls_train, RMSEC_cls, C_cls_cross_validation, RMSECV_cls] = pnnl_napalm_cls
[C_pcr, RMSEP_pcr, C_pcr_train, RMSEC_pcr, C_pcr_cross_validation, RMSECV_pcr] = pnnl_napalm_pcr(nPr
[C_pls, RMSEP_pls, C_pls_train, RMSEC_pls, C_pls_cross_validation, RMSECV_pls] = pnnl_napalm_pls(nLa
```

Optimal number of principal components for PCR with the napalm data

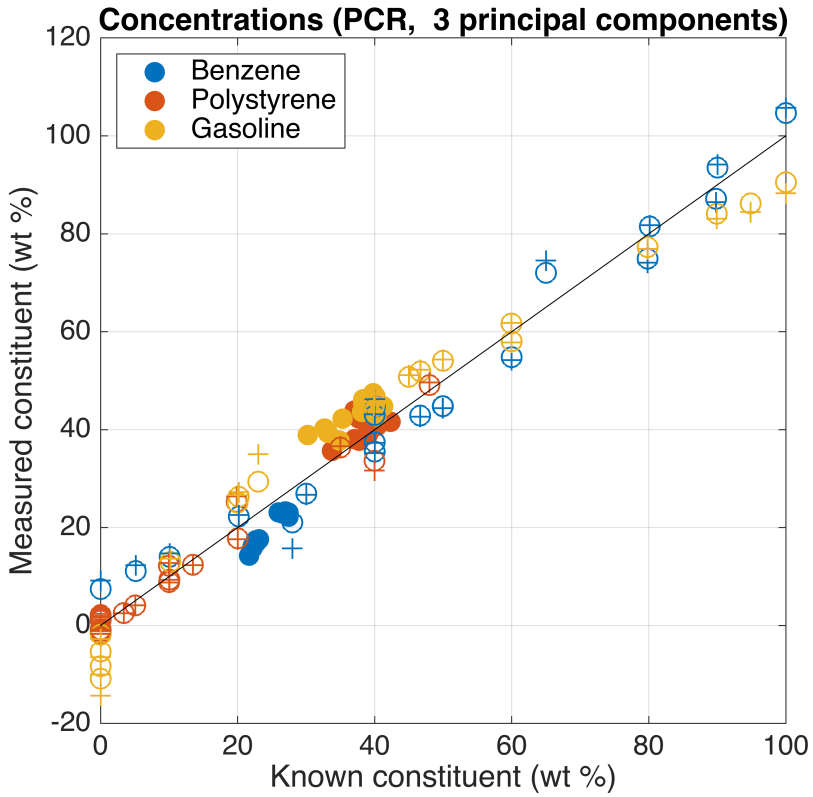
Compute PCR for 1 through 18 principal components and plot RMSEP for them.

```
nPrincipalComponents = 1:18;
[C_pcr, RMSEP_pcr] = pnnl_napalm_pcr(nPrincipalComponents);
plot(nPrincipalComponents, RMSEP_pcr,'.-','LineWidth',2,'MarkerSize',20)
xlabel('Number of principal components')
ylabel('RMSEP')
title('PCR')
legend(ConstituentNames{:})
```



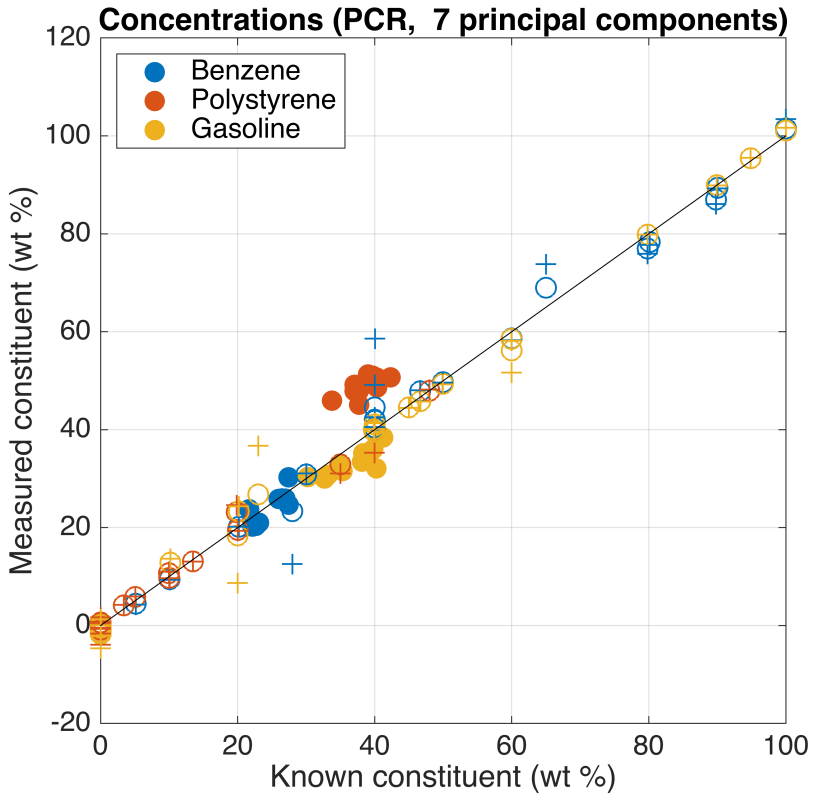
It looks like 3 is the knee in the curve for all constituents, with 7 being better for benzene and 6 being better for gasolene. It may be best to use 3 principal components across the board, but visualize 3, 7, and 6 to see what they look like.

```
nPrincipalComponents = [3 7 6];
pnnl_napalm_pcr(nPrincipalComponents);
```



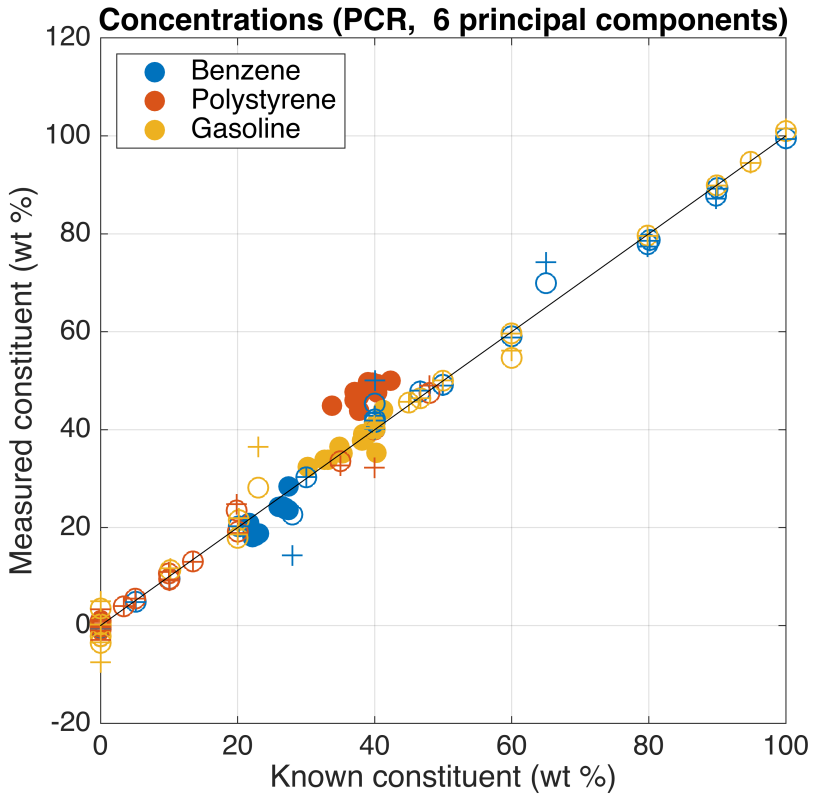
Legend: Dot is predicted. Circle is train. Cross is cross-validation.

PCR, 3 principal components	Benzene	Polystyrene	Gasoline
RMSEC	4.6738	2.3109	5.9859
RMSECV	5.9206	2.8541	7.4667
RMSEP	5.1766	2.7318	6.2906



Legend: Dot is predicted. Circle is train. Cross is cross-validation.

PCR, 7 principal components	Benzene	Polystyrene	Gasoline
RMSEC	2.1605	1.0254	1.7212
RMSECV	6.3531	2.0739	4.7611
RMSEP	1.8345	10.584	3.9805

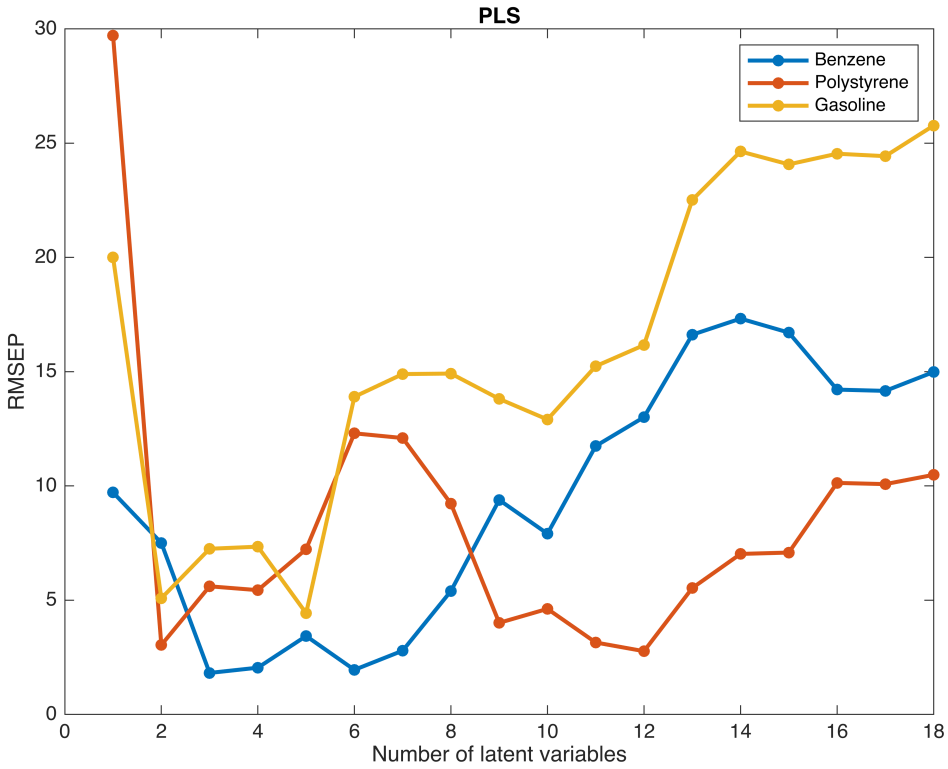



```
Legend: Dot is predicted. Circle is train. Cross is cross-validation.
--------------------------------------------------------------------------
  PCR,  6 principal components       Benzene    Polystyrene       Gasoline
                         RMSEC         2.262         1.1177         2.1723
                        RMSECV         4.499         2.4207         3.8803
                         RMSEP        3.2226         9.2569          1.898
--------------------------------------------------------------------------
```

Optimal number of latent variables for PLS with the napalm data

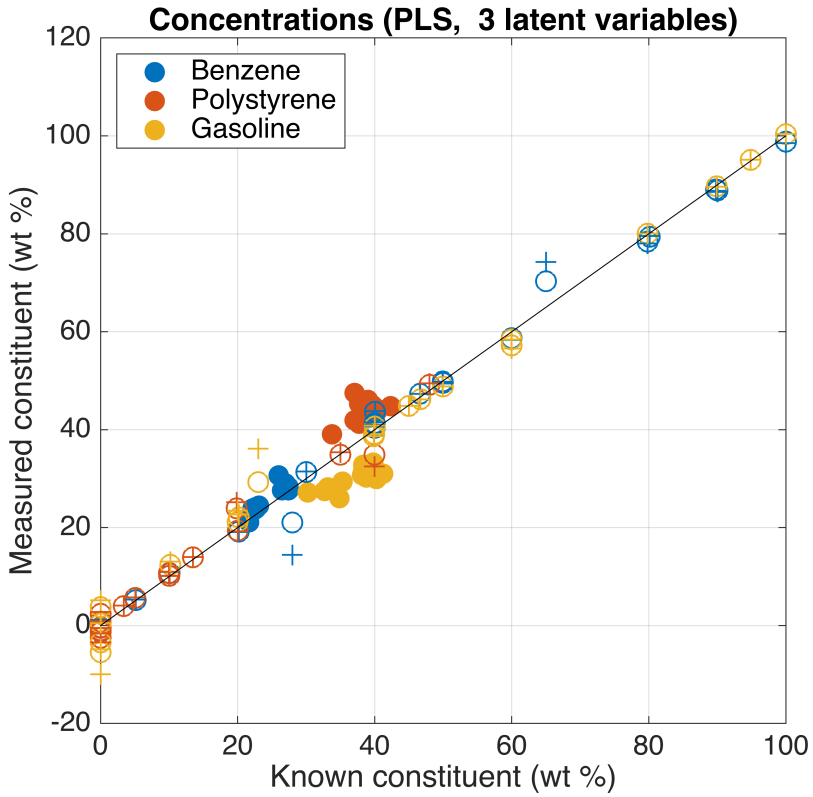
Compute PLS for 1 through 18 latent variables and plot RMSEP for them.

```
nLatentVariables = 1:18;
meanCentered = true;
[C_pls, RMSEP_pls] = pnnl_napalm_pls(nLatentVariables, meanCentered);
plot(nLatentVariables, RMSEP_pls,'.-','LineWidth',2,'MarkerSize',20)
xlabel('Number of latent variables')
ylabel('RMSEP')
title('PLS')
legend(ConstituentNames{:})
```



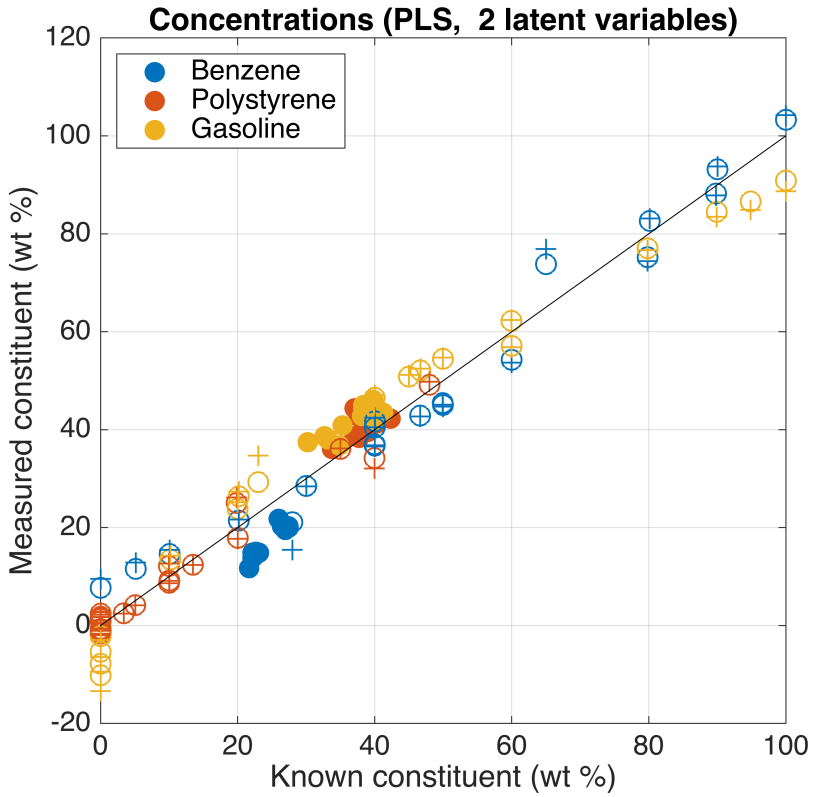
It looks like the knee in the curve for benzene is 3 latent variables, and 2 for polystyrene and gasolene. Plot them to see what they look like.

```
nLatentVariables = [3 2];
meanCentered = true;
pnnl_napalm_pls(nLatentVariables,meanCentered);
```



Legend: Dot is predicted. Circle is train. Cross is cross-validation.

PLS, 3 latent variables	Benzene	Polystyrene	Gasoline
RMSEC	2.2847	1.7662	2.4538
RMSECV	3.9022	2.4136	4.2785
RMSEP	1.8185	5.6092	7.2521




```
Legend: Dot is predicted. Circle is train. Cross is cross-validation.
--------------------------------------------------------------------------
  PLS,  2 latent variables       Benzene    Polystyrene      Gasoline
                    RMSEC        4.5616         2.2064        5.8133
                   RMSECV        5.8788         2.7588        7.2286
                    RMSEP        7.5077           3.04        5.0846
--------------------------------------------------------------------------
```

Single Constituent Analysis

Single-Constituent Analysis Function

Loop over one constituent at a time for a range of number of principal components for PCR or latent variables for PLS

```
function [C_predicted,RMSEP,C_predicted_train,RMSEC,C_cross_validation,RMSECV] = pnnl_single_constitu
    [p_train,n_train] = size(C_train);
    [p_unknown,n] = size(C_validation);
    assert(isequal(n_train,n),'The training and validation sets must have the same number of constitu

    nSamples = length(nComponents);
    C_predicted = zeros(p_unknown, n, nSamples);
    C_predicted_train = zeros(p_train, n, nSamples);
    C_cross_validation = zeros(p_train, n, nSamples);
    for j = 1:n
        for i = 1:nSamples
            r = nComponents(i);
            C_predicted(:,j,i) = method(A_train,C_train(:,j),A_unknown,r);
            C_predicted_train(:,j,i) = method(A_train,C_train(:,j),A_train,r);
```

```
                C_cross_validation(:,j,i) = pnnl_cross_validation(method,A_train,C_train(:,j),r);
            end
        end
        RMSEP  = zeros(nSamples,n);
        RMSEC  = zeros(nSamples,n);
        RMSECV = zeros(nSamples,n);
        for i = 1:nSamples
            RMSEP(i,:)  = pnnl_rmse(C_predicted(:,:,i), C_validation);
            RMSEC(i,:)  = pnnl_rmse(C_predicted_train(:,:,i), C_train);
            RMSECV(i,:) = pnnl_rmse(C_cross_validation(:,:,i), C_train);
        end
    end
```

where

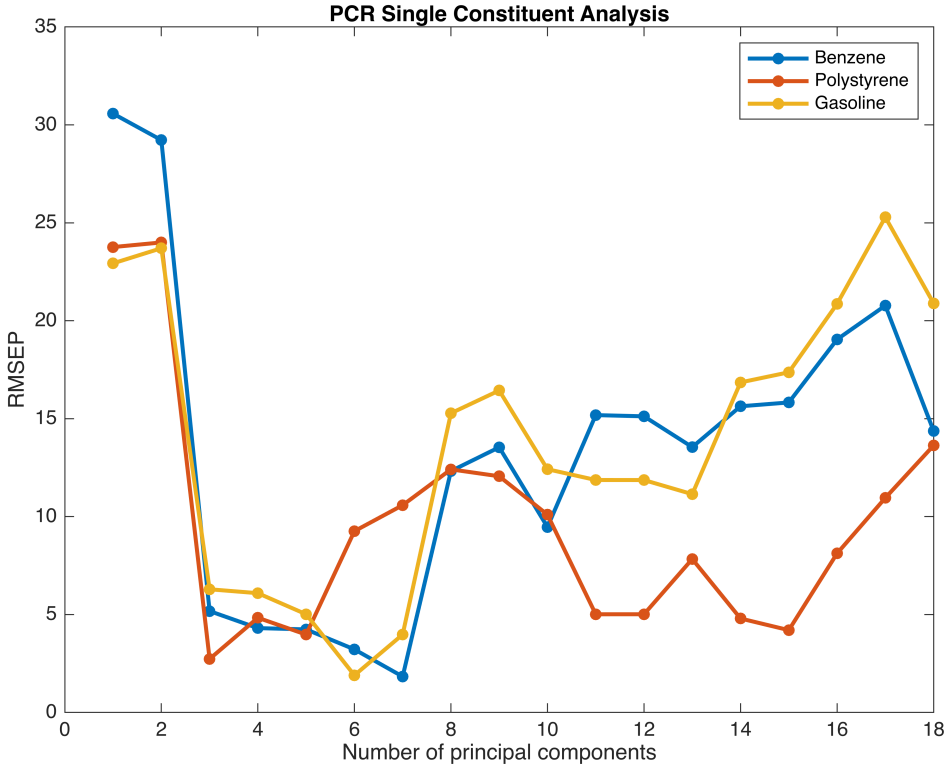
```
    method = @pnnl_pcr
```

or

```
    method = @pnnl_pls
```

PCR single-constituent analysis

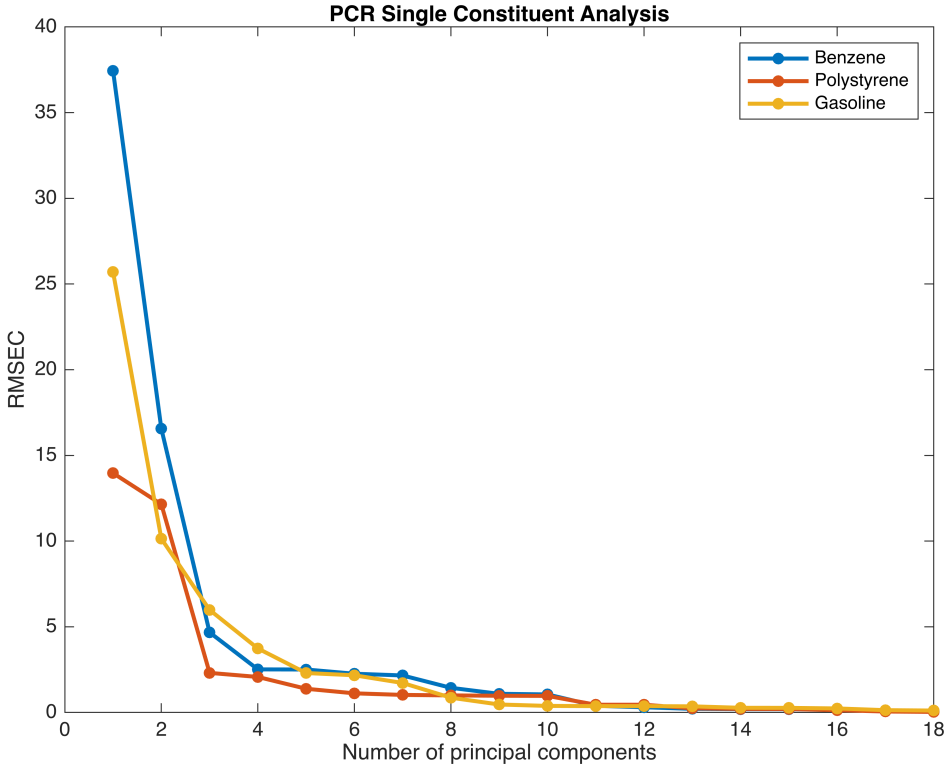
```
nPrincipalComponents = 1:18;
[C_pcr_single,RMSEP_pcr_single,C_pcr_train,RMSEC_pcr_single,C_pcr_cross_validation,RMSE

figure
plot(nPrincipalComponents, RMSEP_pcr_single,'.-','LineWidth',2,'MarkerSize',20)
xlabel('Number of principal components')
ylabel('RMSEP')
title('PCR Single Constituent Analysis')
legend(ConstituentNames{:})
```




```
RMSEP_pcr_single
```

```
RMSEP_pcr_single = 18×3
   30.5754   23.7626   22.9313
   29.2247   23.9980   23.7022
    5.1766    2.7318    6.2906
    4.3080    4.8366    6.0915
    4.2423    3.9788    5.0158
    3.2226    9.2569    1.8980
    1.8345   10.5841    3.9805
   12.3262   12.4126   15.2784
   13.5406   12.0606   16.4394
    9.4668   10.0999   12.4205
     ⋮
```

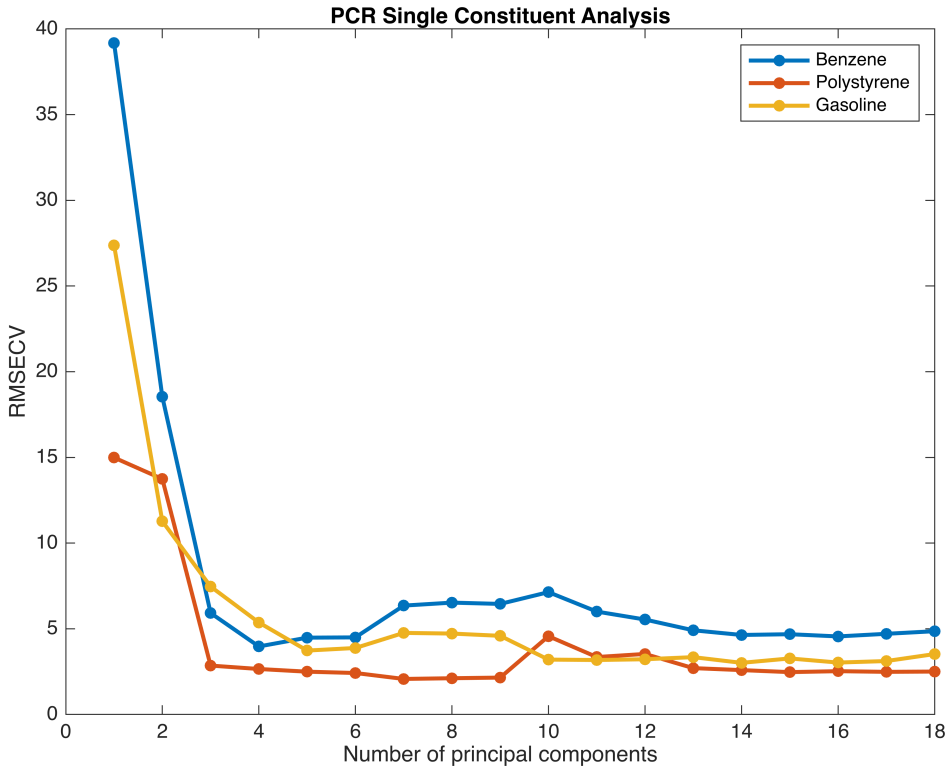
```
figure
plot(nPrincipalComponents, RMSEC_pcr_single,'.-','LineWidth',2,'MarkerSize',20)
xlabel('Number of principal components')
ylabel('RMSEC')
title('PCR Single Constituent Analysis')
legend(ConstituentNames{:})
```




```
RMSEC_pcr_single
```

```
RMSEC_pcr_single = 18×3
   37.4380   13.9664   25.7064
   16.5607   12.1515   10.1419
    4.6738    2.3109    5.9859
    2.5151    2.0734    3.7362
    2.5081    1.3847    2.3022
    2.2620    1.1177    2.1723
    2.1605    1.0254    1.7212
    1.4392    0.9970    0.8625
    1.0894    0.9765    0.4660
    1.0551    0.9680    0.3838
     ⋮
```

```
figure
plot(nPrincipalComponents, RMSECV_pcr_single,'.-','LineWidth',2,'MarkerSize',20)
xlabel('Number of principal components')
ylabel('RMSECV')
title('PCR Single Constituent Analysis')
legend(ConstituentNames{:})
```

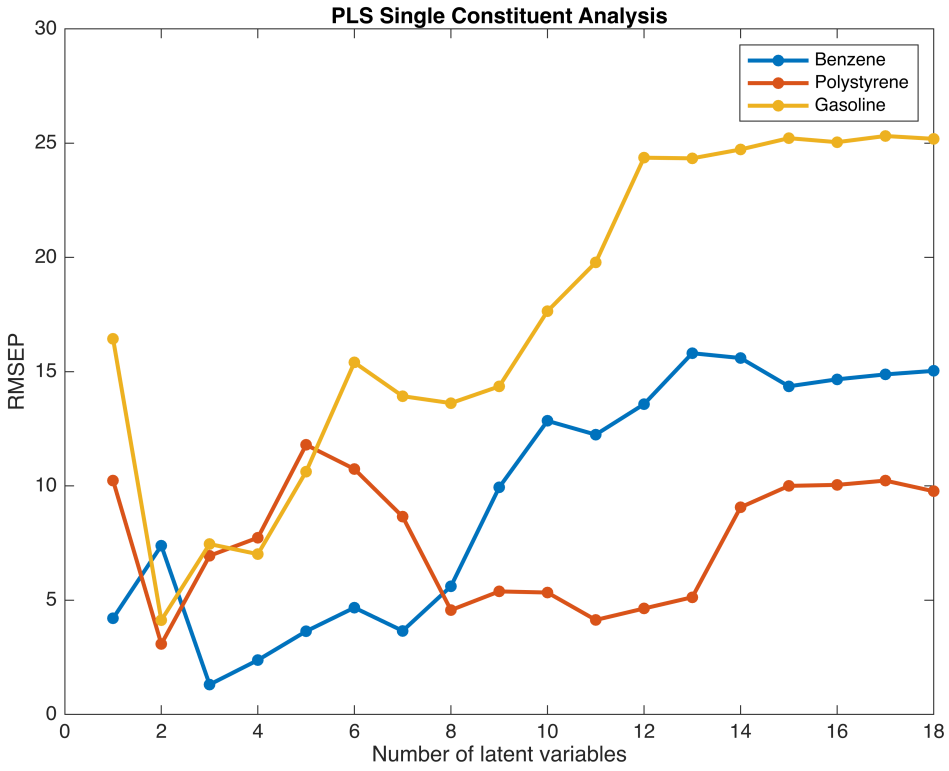



```
RMSECV_pcr_single
```

```
RMSECV_pcr_single = 18×3
   39.1802   14.9894   27.3785
   18.5368   13.7424   11.2736
    5.9206    2.8541    7.4667
    3.9695    2.6545    5.3672
    4.4832    2.4979    3.7294
    4.4990    2.4207    3.8803
    6.3531    2.0739    4.7611
    6.5274    2.1158    4.7216
    6.4514    2.1532    4.5899
    7.1394    4.5678    3.2023
     ⋮
```

PLS single-constituent analysis (PLS1)

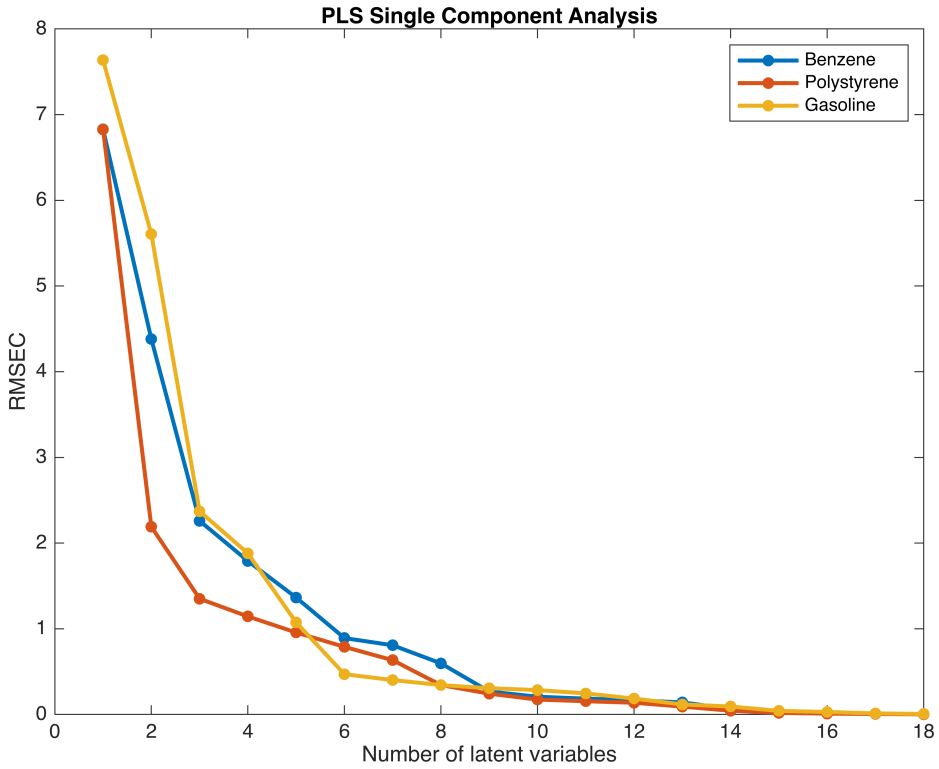
```
nLatentVariables = 1:18;
meanCentered = true;
[C_pls_single,RMSEP_pls_single,C_pls_train,RMSEC_pls_single,C_pls_cross_validation,RMSE

figure
plot(nLatentVariables, RMSEP_pls_single,'.-','LineWidth',2,'MarkerSize',20)
xlabel('Number of latent variables')
ylabel('RMSEP')
title('PLS Single Constituent Analysis')
legend(ConstituentNames{:})
```




```
RMSEP_pls_single
```

```
RMSEP_pls_single = 18×3
    4.2080   10.2402   16.4432
    7.3905    3.0818    4.1248
    1.3096    6.9468    7.4590
    2.3801    7.7360    7.0172
    3.6426   11.8017   10.6290
    4.6758   10.7408   15.4054
    3.6591    8.6560   13.9297
    5.6087    4.5684   13.6261
    9.9386    5.3861   14.3570
   12.8500    5.3368   17.6503
     ⋮
```

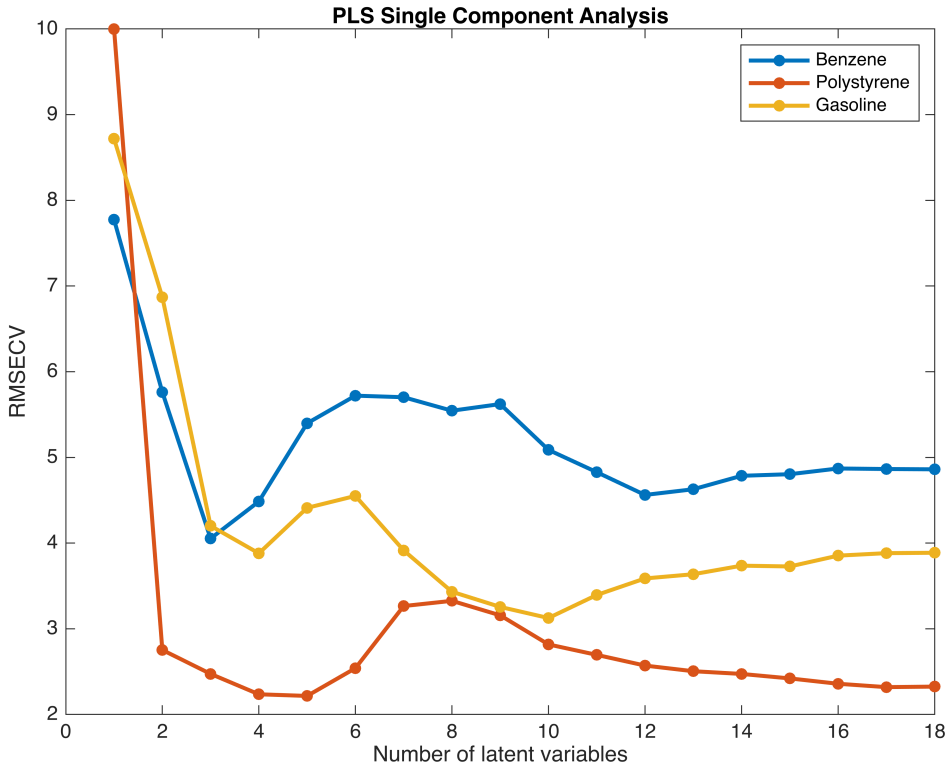
```
figure
plot(nLatentVariables, RMSEC_pls_single,'.-','LineWidth',2,'MarkerSize',20)
xlabel('Number of latent variables')
ylabel('RMSEC')
title('PLS Single Component Analysis')
legend(ConstituentNames{:})
```




```
RMSEC_pls_single
```

```
RMSEC_pls_single = 18×3
    6.8289    6.8274    7.6372
    4.3821    2.1913    5.6067
    2.2597    1.3507    2.3716
    1.7915    1.1442    1.8808
    1.3650    0.9566    1.0751
    0.8919    0.7892    0.4700
    0.8077    0.6338    0.4020
    0.5967    0.3433    0.3430
    0.2679    0.2428    0.3068
    0.2072    0.1740    0.2846
     ⋮
```

```
figure
plot(nLatentVariables, RMSECV_pls_single,'.-','LineWidth',2,'MarkerSize',20)
xlabel('Number of latent variables')
ylabel('RMSECV')
title('PLS Single Component Analysis')
legend(ConstituentNames{:})
```




```
RMSECV_pls_single
```

```
RMSECV_pls_single = 18×3
    7.7757    9.9992    8.7190
    5.7622    2.7529    6.8702
    4.0534    2.4738    4.2017
    4.4860    2.2357    3.8805
    5.3984    2.2166    4.4115
    5.7206    2.5403    4.5507
    5.7025    3.2655    3.9151
    5.5449    3.3264    3.4324
    5.6225    3.1571    3.2554
    5.0880    2.8177    3.1258
     ⋮
```

Principal Components and Principal Component Scores

Scores definition

The scores are the left-singular vectors multiplied by the singular values of the absorbance matrix. The economy-size singular value decomposition is

$$\underset{p\times m}{A} = \underset{p\times p}{U} \cdot \underset{p\times p}{\Sigma} \cdot \underset{p\times m}{V^{\mathrm{T}}}$$

where U and V have orthonormal columns, $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots \sigma_p)$, and $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_p \geq 0$.

It can be written as a sum of rank-one outer products

$$\underset{p\times m}{A} = \sigma_1 \underset{p\times 1}{U}(:,1)\underset{1\times m}{V}(:,1)^{\mathrm{T}} + \sigma_2 \underset{p\times 1}{U}(:,2)\underset{1\times m}{V}(:,2)^{\mathrm{T}} + \cdots + \sigma_p \underset{p\times 1}{U}(:,p)\underset{1\times m}{V}(:,p)^{\mathrm{T}}.$$

where $\sigma_i U(:,i)$ is called the score of the i th principal component $V(:,i)$ (Cleve Moler, "Professor SVD").

In matrix form, this is written as

$$\text{Scores} = U\Sigma.$$

Singular Value Decomposition

Function `pnnl_rectifiedSVD` returns the economy-size singular value decomposition, where the maximum magnitude element of each column of V is positive, which is the convention for principal components. The singular value decomposition is unique up to the signs of the columns of U and V, like $1*2*1 == -1*2*-1$. Rectified singular value decompositions are unique, which makes it convenient to compare one score to another.

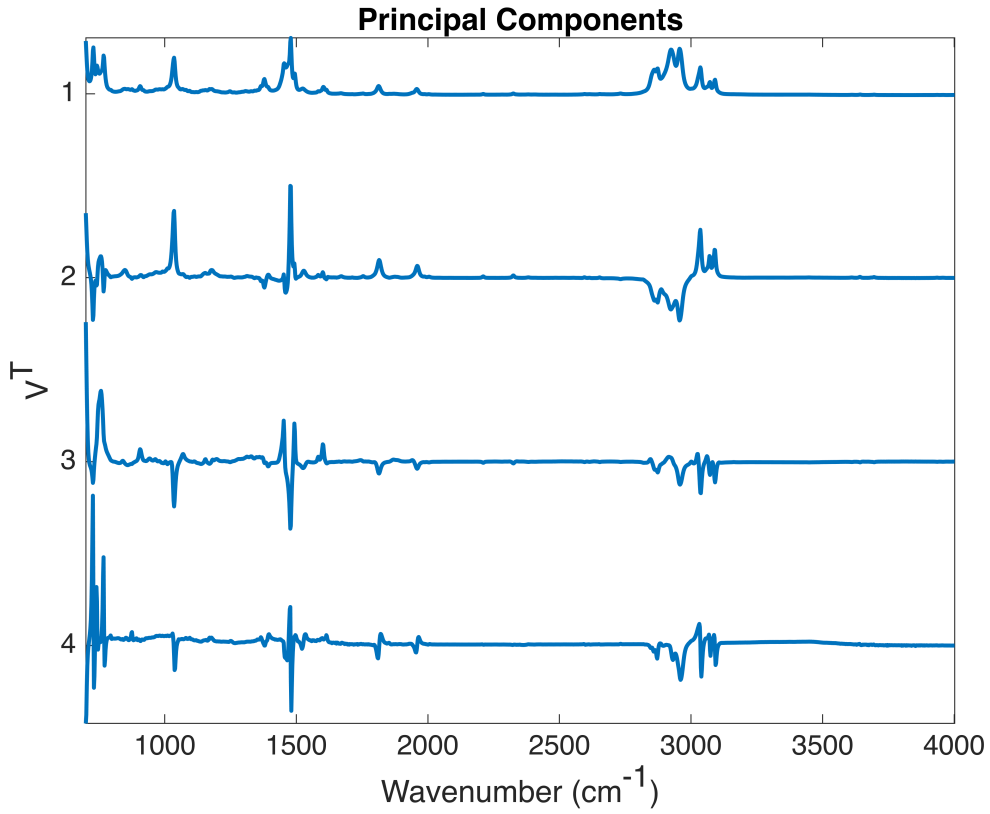
```
[U,S,V] = pnnl_rectifiedSVD(A_train);
```

Principal Components

The principal components are the right singular vectors in the columns of matrix V.

Plot the first four principal components.

```
figure
pnnl_strip_plot(Wavenumbers,V(:,1:4)',...
    'Principal Components',...
    WavenumberLabel,'V^T')
```



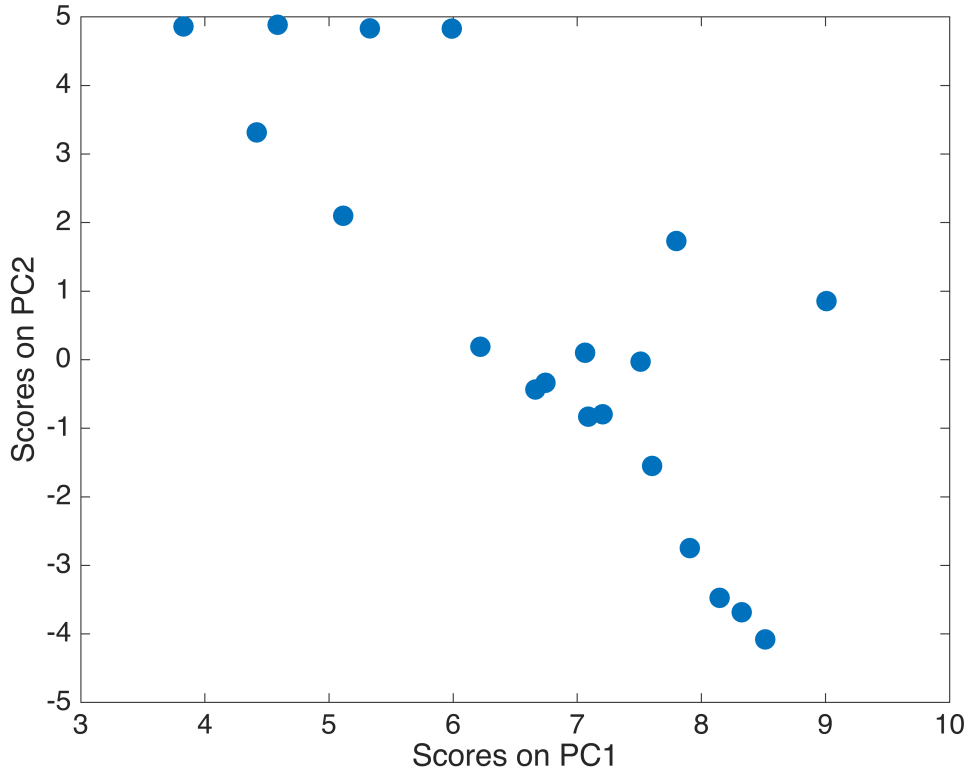
Principal Compent Scores

The scores are the left singular vectors multplied by the singular values.

```
Scores = U*S;
```

Plot the scores of one principal component against the other.

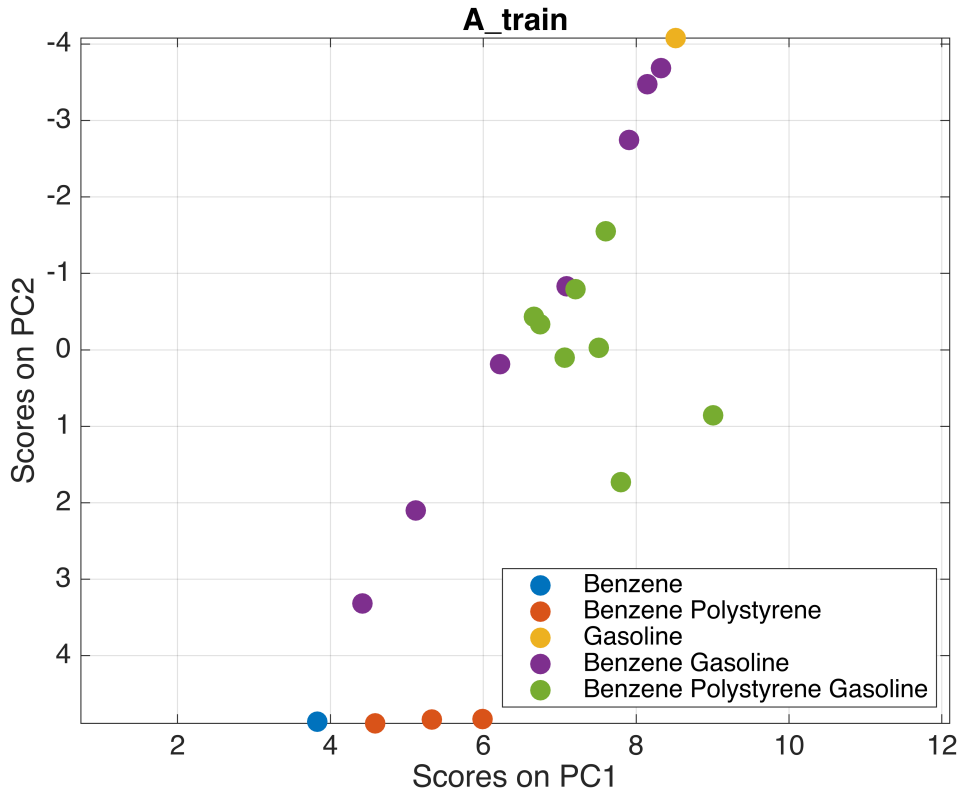
```
fontSize = 14;
plot(Scores(:,1),Scores(:,2),'.','MarkerSize',35);
xlabel('Scores on PC1');
ylabel('Scores on PC2');
set(gca,'FontSize',fontSize);
box on
```



Plot scores with labels

The scores only rely on an absorbance matrix, without having to know the actual concentrations. If you also know the concentrations, then you can use that information to label the scores. Function `pnnl_plot_scores` computes and plots the scores as in the previous section, and also uses information in the known concentrations to label the scores plot.

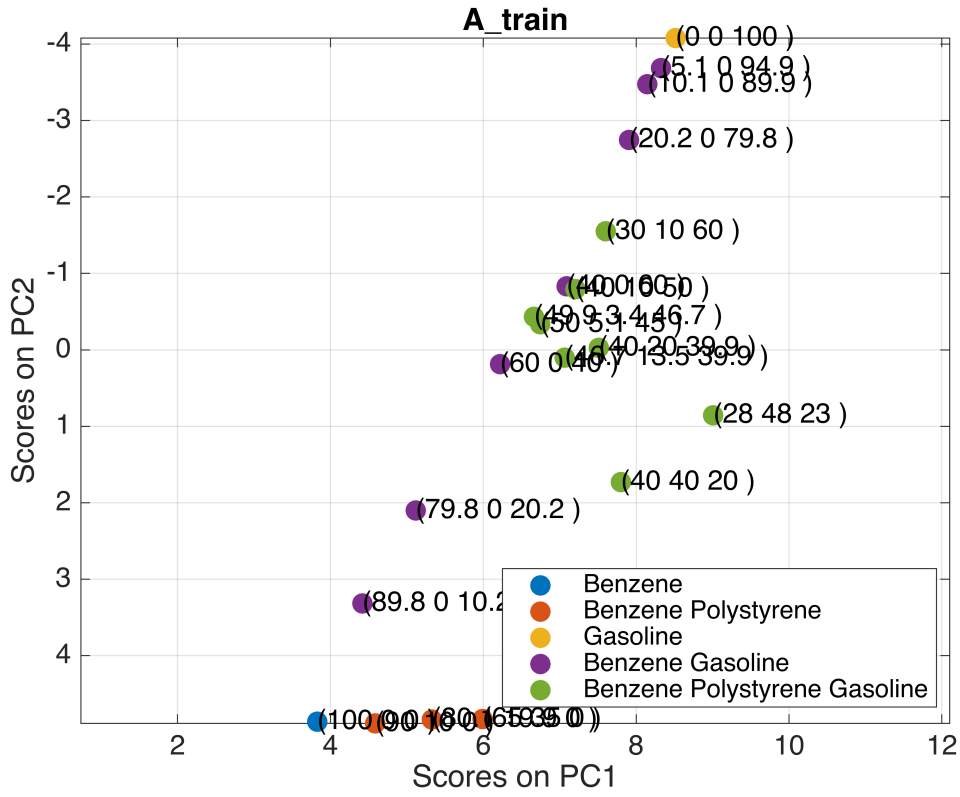
```
pnnl_plot_scores(A_train, C_train, 1, 2, ConstituentNames)
```



Display concentration percentages of training set

If you input an additional parameter to `pnnl_plot_scores`, then it will also display the concentration percentages as (%benzene, %polystyrene, %gasoline), which may be useful to zoom in and explore the linear relationship between the scores.

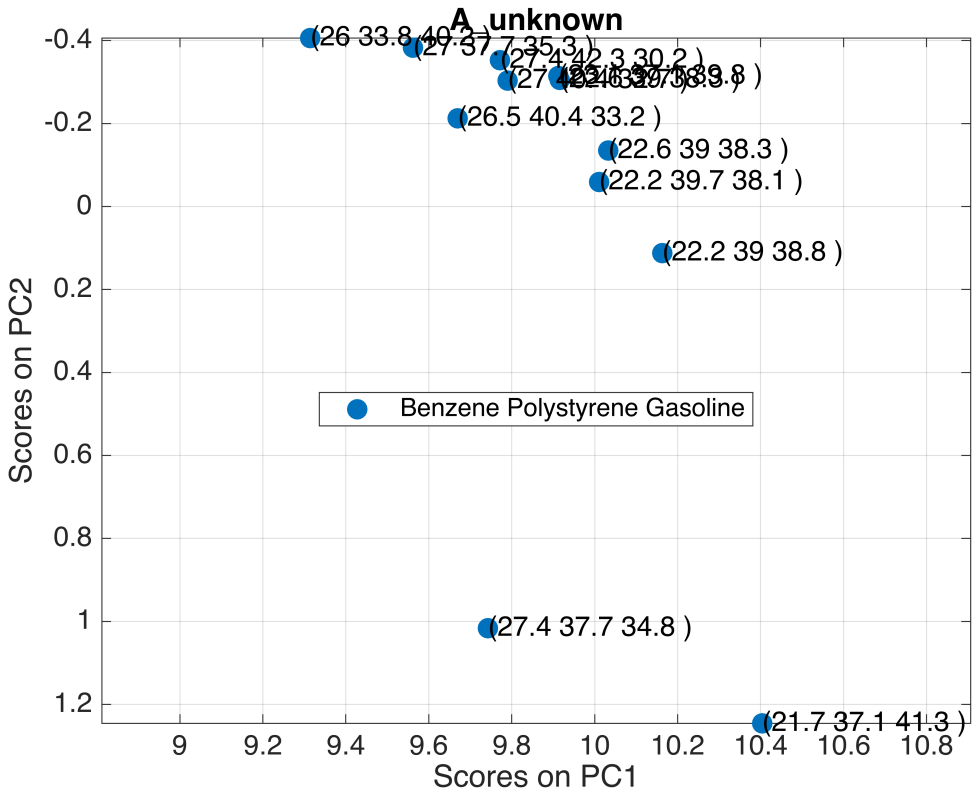
```
displayConcentrations = true;
pnnl_plot_scores(A_train, C_train, 1, 2, ConstituentNames, displayConcentrations)
```



Display concentration percentages of unknown set

If you input an additional parameter to `pnnl_plot_scores`, then it will also display the concentration percentages as (%benzene, %polystyrene, %gasoline), which may be useful to zoom in and explore the linear relationship between the scores.

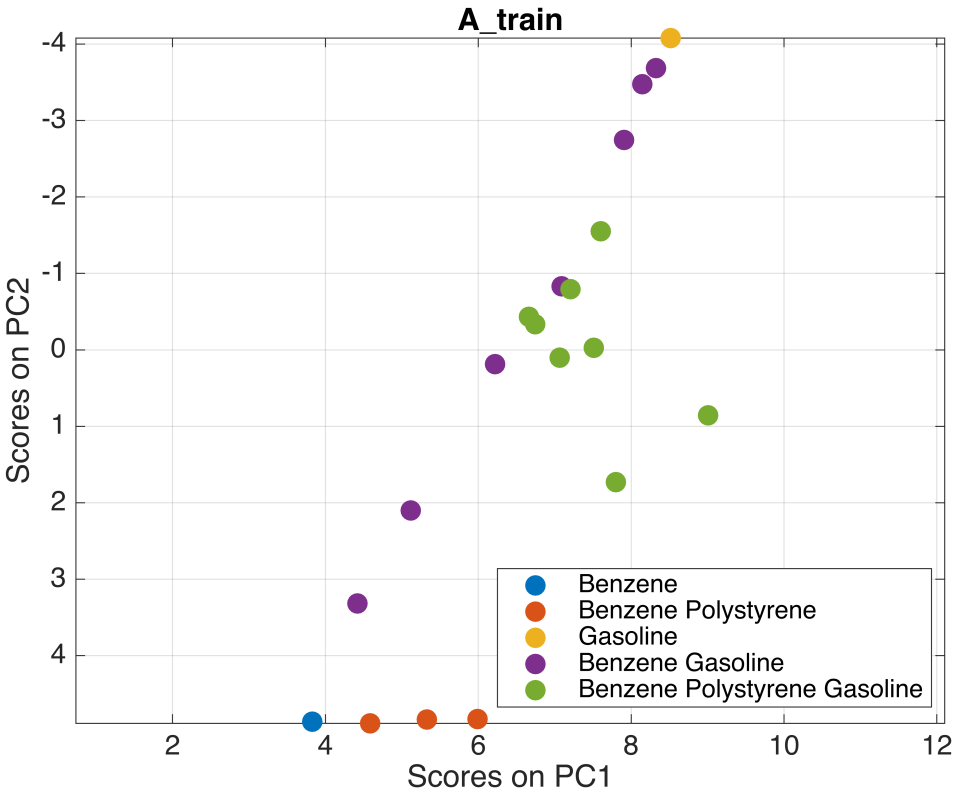
```
displayConcentrations = true;
pnnl_plot_scores(A_unknown, C_validation, 1, 2, ConstituentNames, displayConcentrations
```

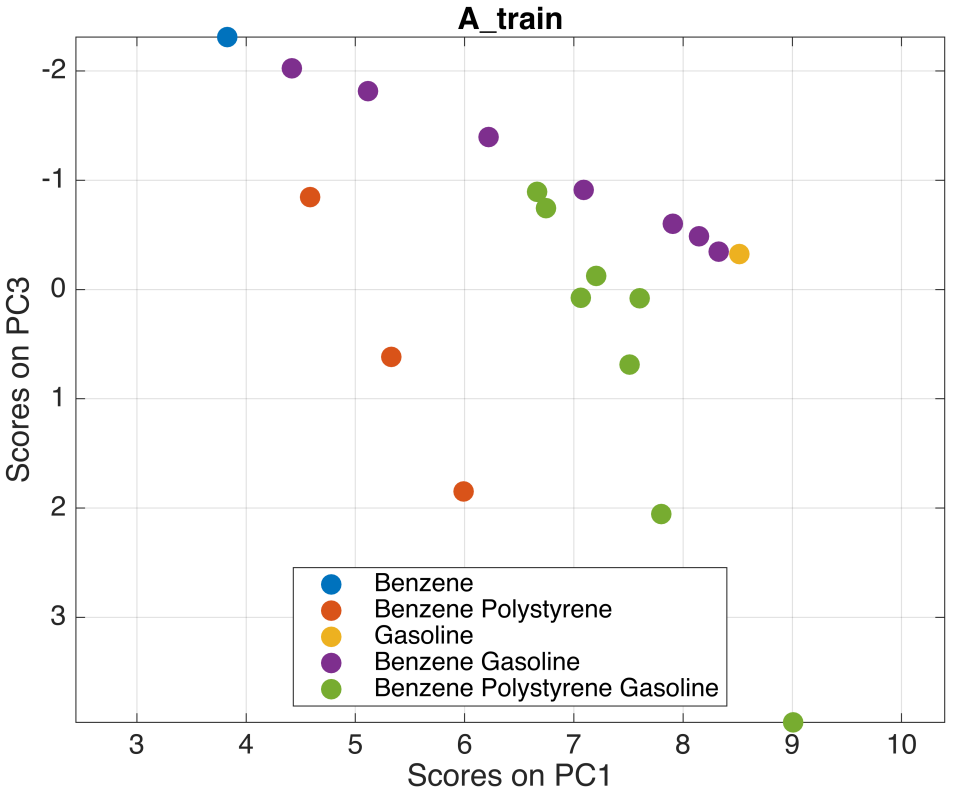


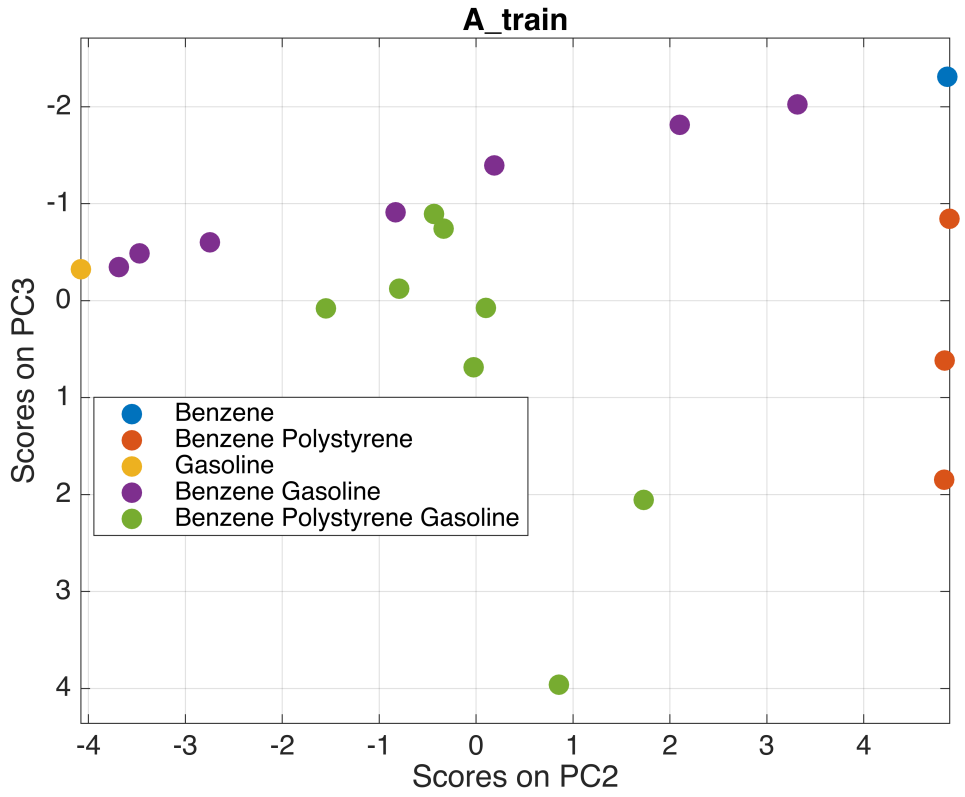
Display all combinations of scores

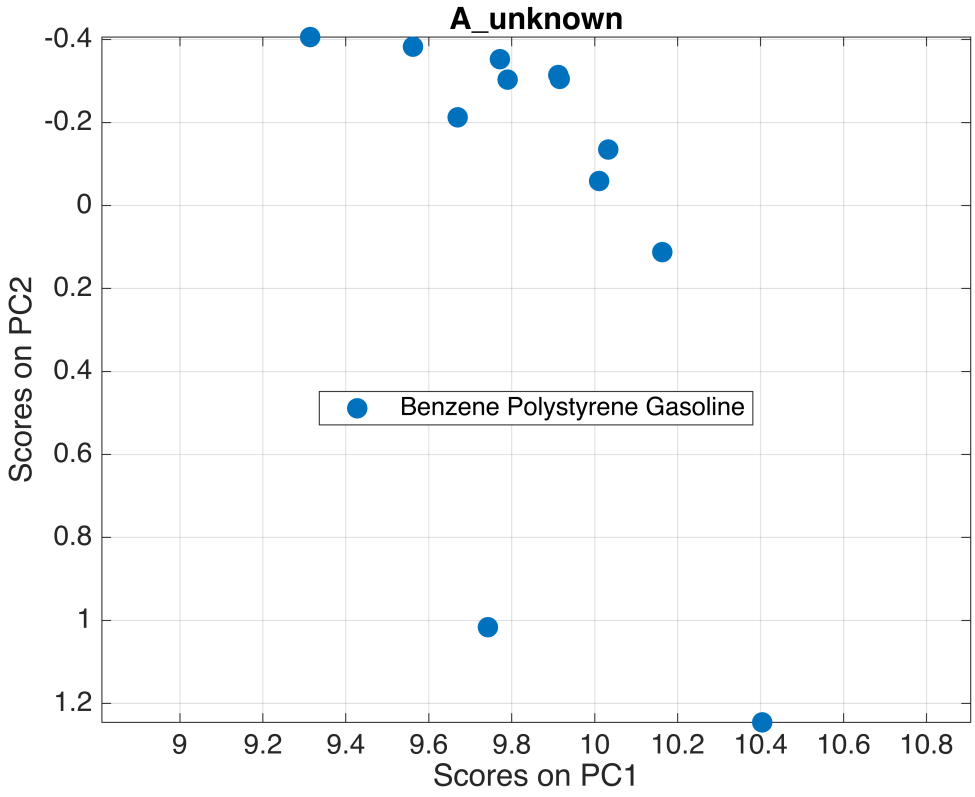
Function `pnnl_napalm_plot_scores` loads the napalm data set and loops over all combinations of principal components.

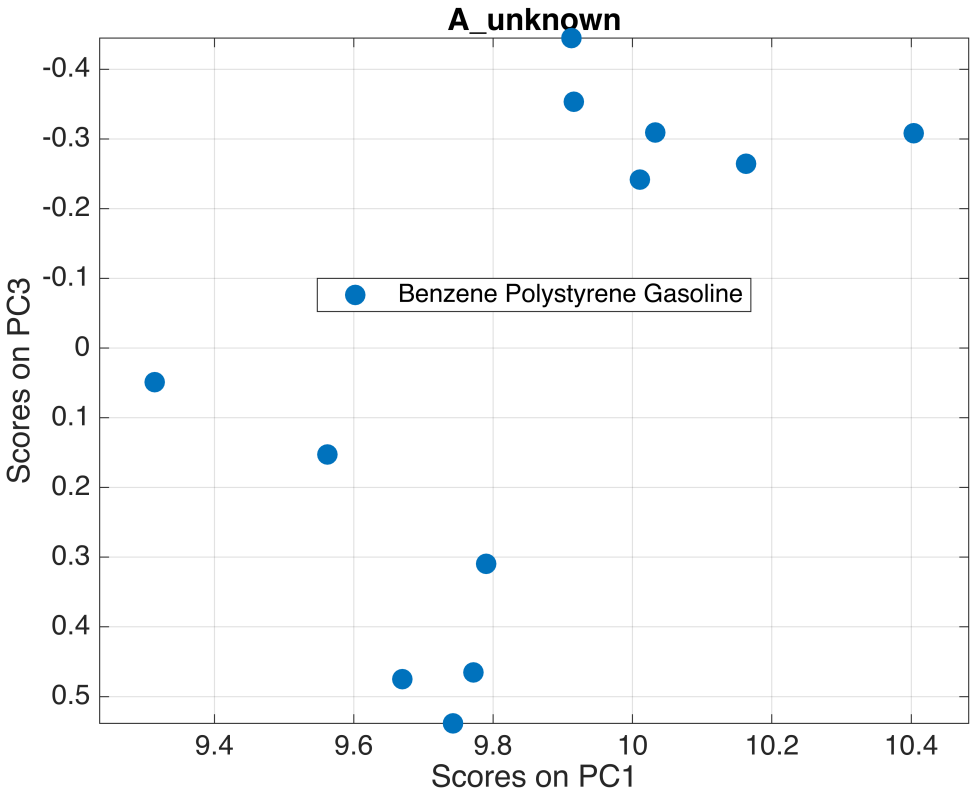
```
pnnl_napalm_plot_scores
```

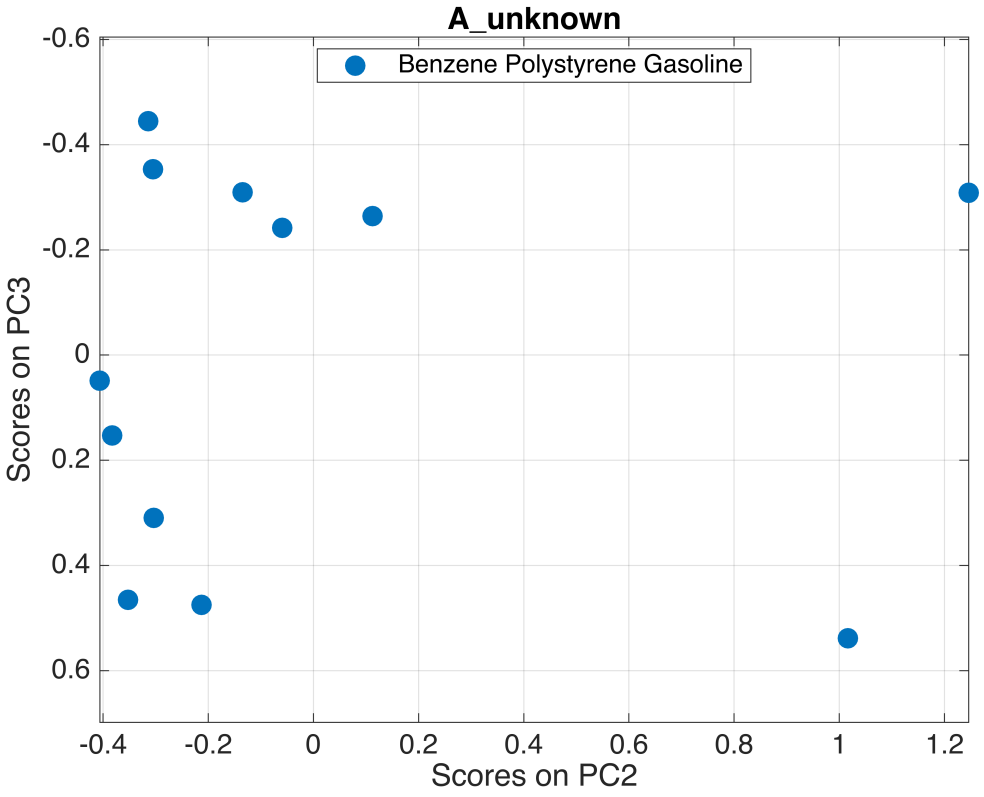

A_train
Scores on PC2
Scores on PC1
Benzene
Benzene Polystyrene
Gasoline
Benzene Gasoline
Benzene Polystyrene Gasoline


A_train
Scores on PC3
Scores on PC1
Benzene
Benzene Polystyrene
Gasoline
Benzene Gasoline
Benzene Polystyrene Gasoline


A_train
Scores on PC3
Scores on PC2
Benzene
Benzene Polystyrene
Gasoline
Benzene Gasoline
Benzene Polystyrene Gasoline


A_unknown
Scores on PC2
Scores on PC1
Benzene Polystyrene Gasoline


A_unknown
Benzene Polystyrene Gasoline
Scores on PC3
Scores on PC1
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
9.4
9.6
9.8
10
10.2
10.4

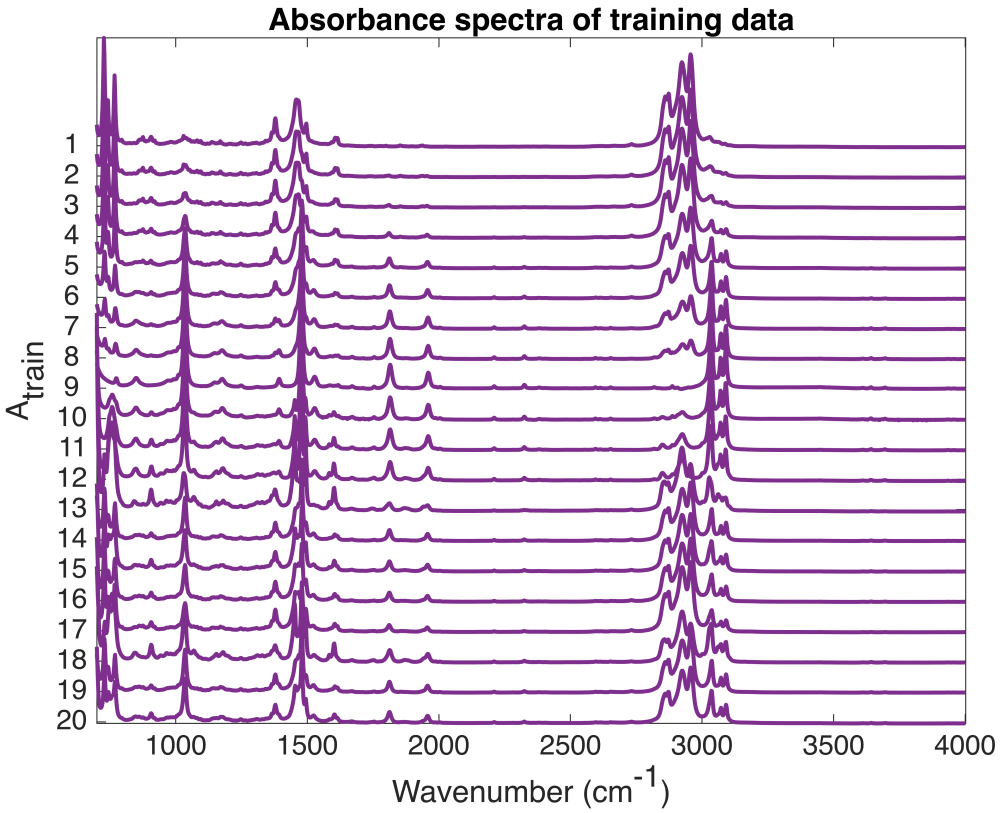

A_unknown
Benzene Polystyrene Gasoline
Scores on PC3
Scores on PC2
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
1.2

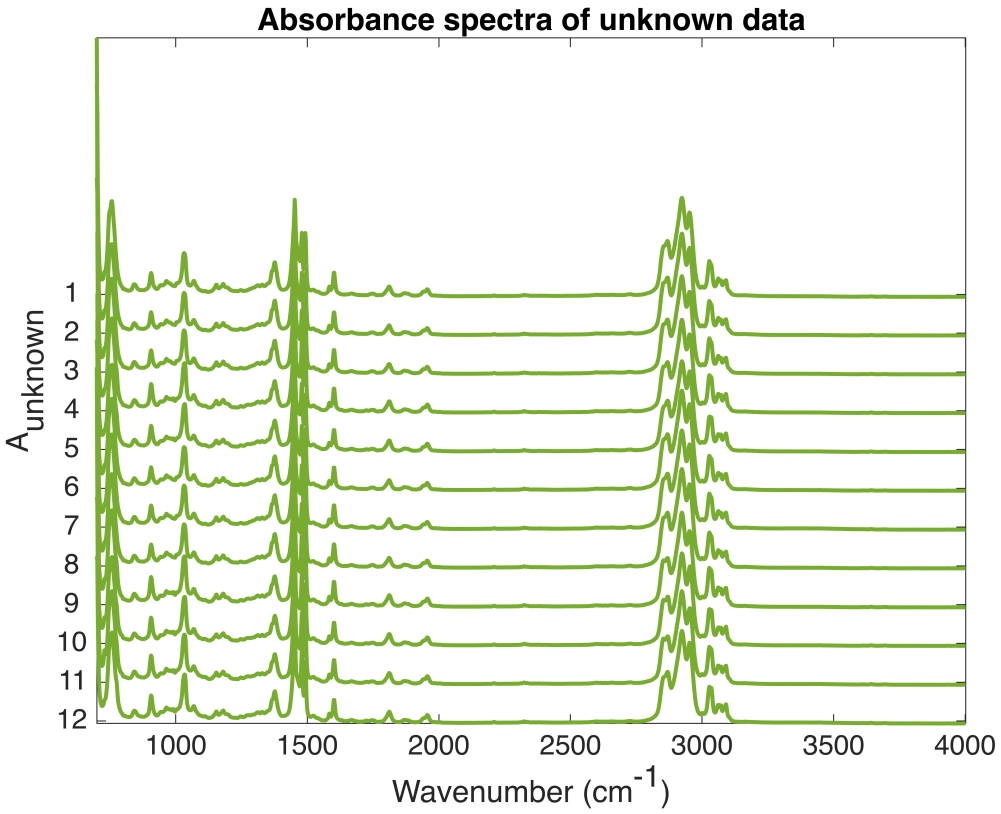
Spectra

Measured spectra

The `pnnl_napalm_plot_spectra` function plots the napalm spectra meaasured in A_{train} and A_{unknown}.

```
pnnl_napalm_plot_spectra
```





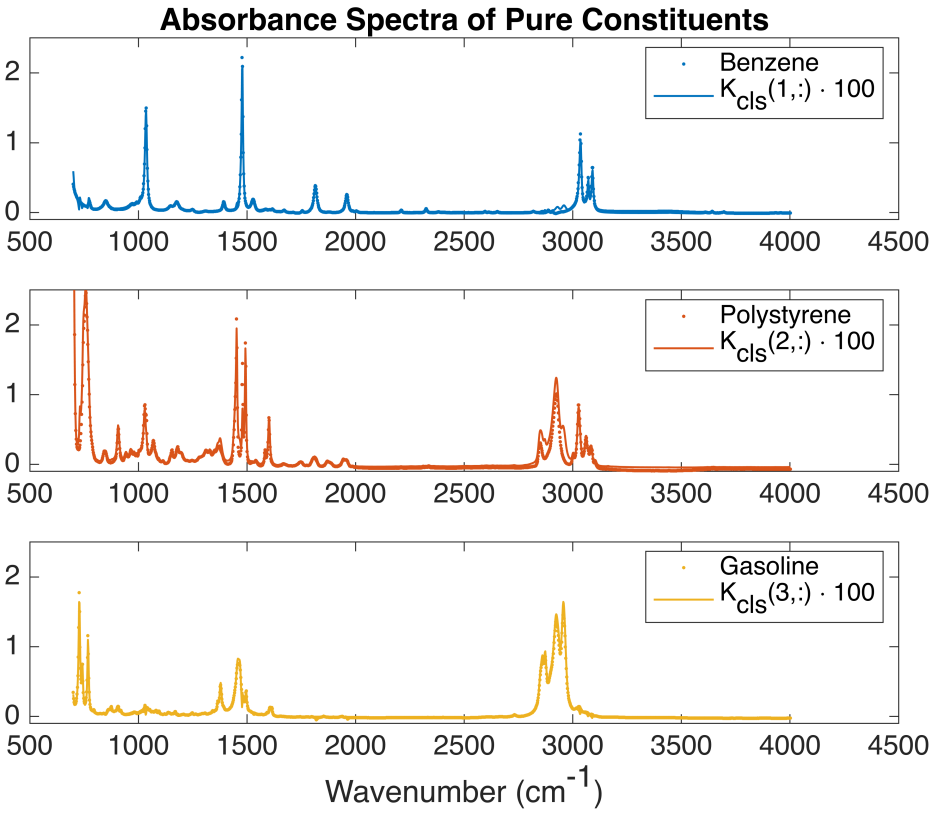
CLS K vs Pure Spectra

Classical least squares computes C_{cls} that minimizes $||CK - A||_2$, where $CK = A$ is the Beer's law relationship between concentration C, extinction coefficient K and absorbtion A. An estimation of the extinction coefficient matrix K_{cls} is computed as a byproduct of the CLS method.

```
function [C_cls, K_cls] = pnnl_cls(A_train, C_train, A_unknown)
    K_cls = C_train \ A_train;
    C_cls = A_unknown / K_cls;
end
```

You can see in the following plots that the rows of K_{cls} match the spectra of the pure constituents.

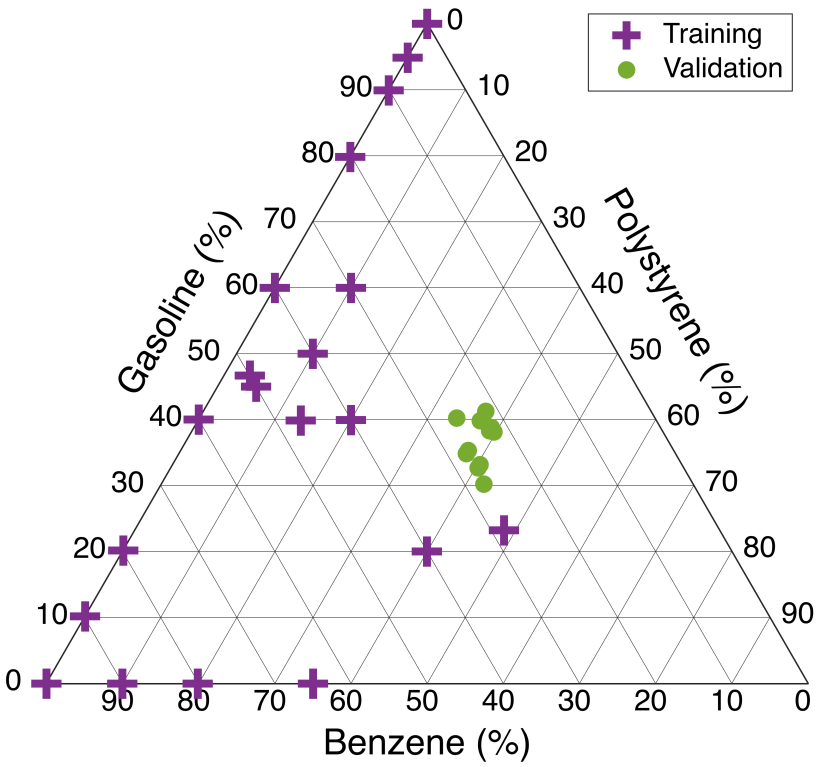
```
[C_cls, B_cls, K_cls] = pnnl_cls(A_train, C_train, A_unknown);
pnnl_napalm_plot_pure_spectra_vs_K_cls
```



Ternary plot

You can see the overlap between C_{train} and $C_{\text{validation}}$ concentraions by plotting them in a ternary plot.

```
pnnl_napalm_plot_concentration_ternary
```



Glossary

Methods

- Classical Least Squares (CLS)
- Principal Component Regression (PCR)
- Partial Least Squares (PLS)

Numbers

- m is the number of measured wavenumbers. In the napalm data, $m=1713$.
- n is the number of constituents. In the napalm data, $n = 3$ for constituents benzene, polystyrene, and gasoline.
- p is the number of compounds. In the napalm data, $p_{\text{train}} = 20$ for the training compounds, and $p_{\text{unknown}} = 12$ for the unknown compounds.
- r is the number of principal components in PCR, and the number of latent variables in PLS.

Matrices

- A is a matrix of absorbance spectra
- C is a matrix of concentrations
- K is a matrix of normalized extinction coefficients

Specific Matrices

- A_{train} is the measured absorbance matrix from the training solutions.
- A_{unknown} is the measured absorbance matrix from the solution with unknown concentrations
- C_{train} is the matrix of known concentrations (measured into the solution)
- $C_{\text{validation}}$ is the matrix of concentrations of the unknown solution derived by an independent means
- C_{cls}, C_{pcr}, C_{pls} is the concentration matrix of the unknown solution computed using CLS, PCR, PLS respectively
- K_{cls}, K_{pcr}, K_{pls} is the matrix of normalized extinction coefficients computed using CLS, PCR, PLS respectively

Root Mean Square Error

- $\text{RMSE} = \text{pnnl\_rmse}(C_{\text{computed}}, C_{\text{known}}) = \sqrt{\text{mean}\left((C_{\text{computed}} - C_{\text{known}}).^2\right)}$. Root mean square error between a computed value and a known value.
- $\text{RMSEP} = \text{pnnl\_rmse}(C_{\text{predicted}}, C_{\text{validation}})$. RMSEP is the prediction error measured on real cases and compared to reference values.
- $\text{RMSEC} = \text{pnnl\_rmse}(C_{\text{predicted from train}}, C_{\text{train}})$. RMSEC is the calibration error. $C_{\text{predicted from train}}$ is computed by using A_{train} as the unknown.
- $\text{RMSECV} = \text{pnnl\_rmse}(C_{\text{cross validation}}, C_{\text{train}})$. RMSECV is the cross-validation error. For each of the rows in the training set C_{train} and A_{train}, leave one row out, and use that row as the unknown data in the predictor. Store the result in $C_{\text{cross-validation}}$. Then compute the RMSE between $C_{\text{cross validation}}$ and C_{train}.

Disclaimer

BATTELLE

for the

UNITED STATES DEPARTMENT OF ENERGY

under Contract DE-AC05-76RL01830