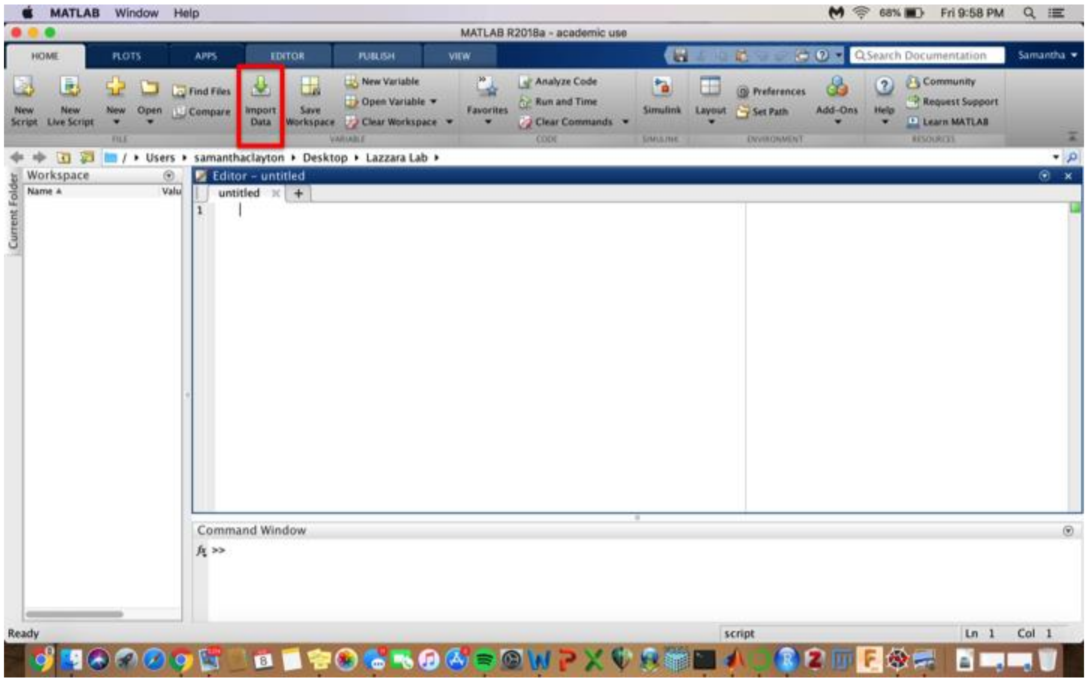
Lazzara Lab
Samantha Clayton
April 14, 2019

Partial Least-Squares Regression (PLSR) in MATLAB R2018a

Importing Data into MATLAB

1. Click on the Home tab in Matlab. Press the "Import Data" button and select the dataset you would like to use.

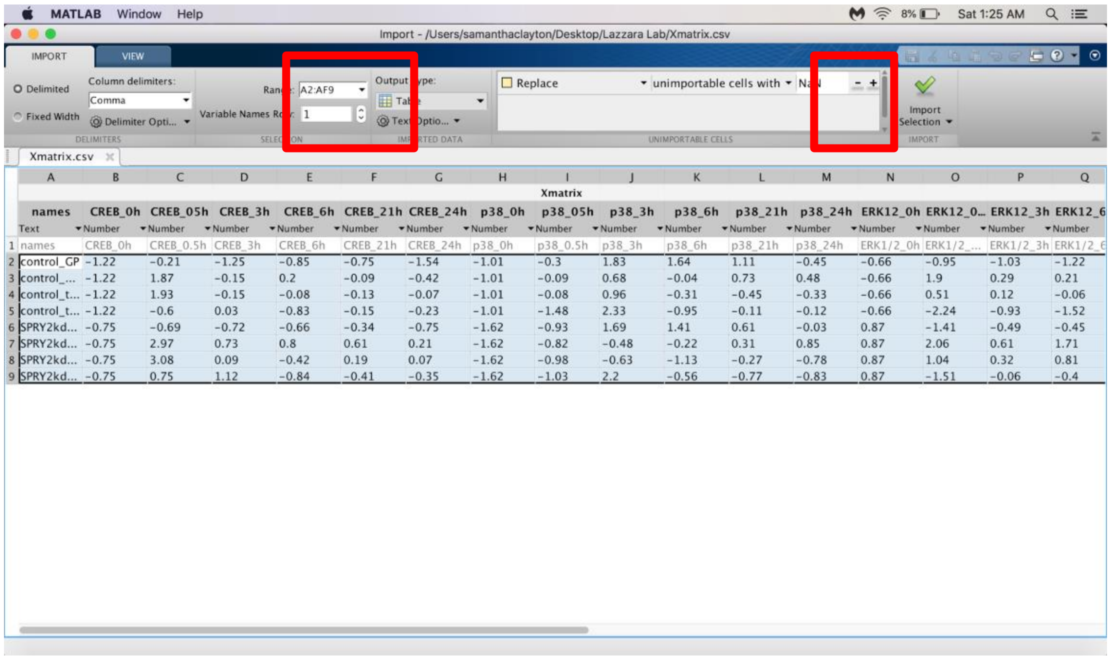


2. The dataset will open onto a screen. Select the data you would like to use then press the "Import Selection" button. The output type options are table, column vectors, numeric matrix, cell array, or string array depending on what you are importing. I suggest using a string array for this process to retain the column labels (if your dataset has them) and to more easily manipulate them. If you select something other than a table, make sure that all the rows and columns you wish to import are selected, as MATLAB may not select them by default.

    a. You can also click on the arrow on the Import Selection button to get a drop-down menu. You can click on generate script or function to auto-import this if you need to run it again.
    b. Or you could save the workspace after importing all of the files you need and load the workspace rather than having to import the data again if you need to run this a separate time.

```
save(plsr_workspace) % saves workspace as a .mat file with the given filename
load(plsr_workspace.mat) % loads workspace into MATLAB
```

Lazzara Lab
Samantha Clayton
April 14, 2019



Import - /Users/samanthaclayton/Desktop/Lazzara Lab/Xmatrix.csv

Delimited; Column delimiters: Comma; Range: A2:AF9; Variable Names Row: 1; Output Type: Table; Replace unimportable cells with NaN

Xmatrix.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	names	CREB_0h	CREB_05h	CREB_3h	CREB_6h	CREB_21h	CREB_24h	p38_0h	p38_05h	p38_3h	p38_6h	p38_21h	p38_24h	ERK12_0h	ERK12_0...	ERK12_3h	ERK12_6
	Text	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
1	names	CREB_0h	CREB_0.5h	CREB_3h	CREB_6h	CREB_21h	CREB_24h	p38_0h	p38_0.5h	p38_3h	p38_6h	p38_21h	p38_24h	ERK1/2_0h	ERK1/2_...	ERK1/2_3h	ERK1/2_6
2	control_GP	-1.22	-0.21	-1.25	-0.85	-0.75	-1.54	-1.01	-0.3	1.83	1.64	1.11	-0.45	-0.66	-0.95	-1.03	-1.22
3	control_...	-1.22	1.87	-0.15	0.2	-0.09	-0.42	-1.01	-0.09	0.68	-0.04	0.73	0.48	-0.66	1.9	0.29	0.21
4	control_t...	-1.22	1.93	-0.15	-0.08	-0.13	-0.07	-1.01	-0.08	0.96	-0.31	-0.45	-0.33	-0.66	0.51	0.12	-0.06
5	control_t...	-1.22	-0.6	0.03	-0.83	-0.15	-0.23	-1.01	-1.48	2.33	-0.95	-0.11	-0.12	-0.66	-2.24	-0.93	-1.52
6	SPRY2kd...	-0.75	-0.69	-0.72	-0.66	-0.34	-0.75	-1.62	-0.93	1.69	1.41	0.61	-0.03	0.87	-1.41	-0.49	-0.45
7	SPRY2kd...	-0.75	2.97	0.73	0.8	0.61	0.21	-1.62	-0.82	-0.48	-0.22	0.31	0.85	0.87	2.06	0.61	1.71
8	SPRY2kd...	-0.75	3.08	0.09	-0.42	0.19	0.07	-1.62	-0.98	-0.63	-1.13	-0.27	-0.78	0.87	1.04	0.32	0.81
9	SPRY2kd...	-0.75	0.75	1.12	-0.84	-0.41	-0.35	-1.62	-1.03	2.2	-0.56	-0.77	-0.83	0.87	-1.51	-0.06	-0.4

3. The dataset will be imported into MATLAB as the data output type you selected with the same name as the original file. If the data is in two separate files, repeat the previous steps for the second file. The remaining steps and sample code outlined below are written assuming that the data is imported using <u>string arrays</u>. Separate the data and labels:

```
X = str2double(Xmatrix(2:end, 2:end));
Y = str2double(Ymatrix(2:end, 2:end));
X_col_labels = cellstr(Xmatrix(1,2:end));
X_row_labels = cellstr(Xmatrix(2:end,1));
Y_col_labels = cellstr(Ymatrix(1,2:end));
Y_row_labels = cellstr(Ymatrix(2:end,1));
```

4. If x and y data are in the same table, allocate the x and y portions of the array into two separate arrays:

```
Data = str2double(data_matrix(2:end,2:end));
X = Data(:,1:last_x_col)
Y = Data(:,first_y_col:end)  %last_x_col and first_y_col need to be defined based on your input
X_col_labels = str2cell(Xmatrix(1,2:end));
Y_col_labels = X_row_labels;
X_row_labels(data_matrix(2:last_x_col,1)
Y_row_labels(data_matrix(first_y_col:end,1)
```

Performing the Regression

5. The data needs to be normalized before using PLSR. You can use the *zscore* function to do this.

```
z_x = zscore(X);
z_y = zscore(Y);
```

6. Code for PSLR function:

```
ncomp = 5; %use only 5 components → change this to desired number of components
[x_loadings, y_loadings, x_scores, y_scores, beta, pctvar, mse, stats] = plsregress(z_x, z_y, ncomp);
```

- If there are any missing values in the matrix, they must be removed before proceeding to this step.
- If ncomp is omitted, its default value is min(size(X,1)-1,size(X,2)).
- You can also specify parameter name/value pairs for cross-validation, ‘cv’ or ‘mcreps’ (See step 10 for sample code used to perform cross-validation).

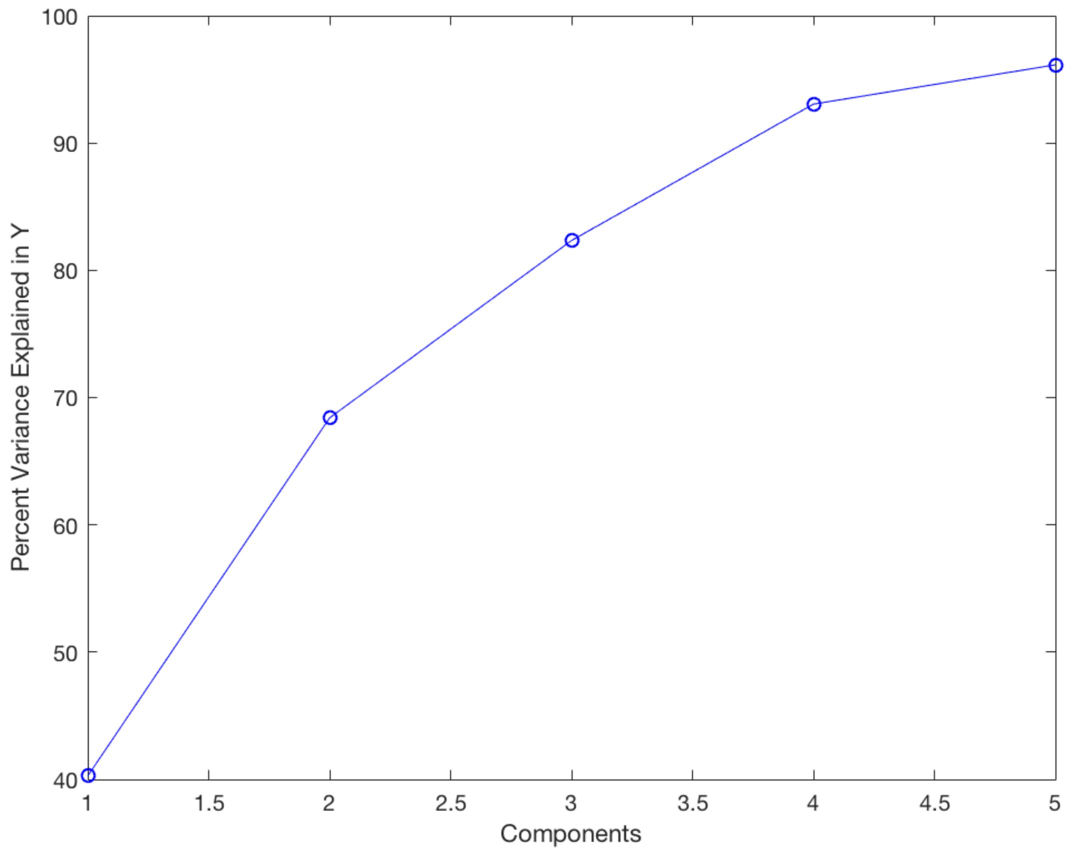
From the MathWorks documentation for *plsregress*:

- *Plsregress* computes a partial least-squares (PLS) regression of Y on X, using ncomp PLS components, and returns the predictor and response loadings in XL and YL, respectively.
- ***X*** is an n-by-p matrix of predictor variables, with rows corresponding to observations and columns to variables.
- ***Y*** is an n-by-m response matrix.
- ***XL*** is a p-by-ncomp matrix of predictor loadings, where each row contains coefficients that define a linear combination of PLS components that approximate the original predictor variables.
- ***YL*** is an m-by-ncomp matrix of response loadings, where each row contains coefficients that define a linear combination of PLS components that approximate the original response variables.
- ***XS*** represents the predictor scores, that is, the PLS components that are linear combinations of the variables in X. XS is an n-by-ncomp orthonormal matrix with rows corresponding to observations and columns to components.
- ***YS*** represents the linear combinations of the responses with which the PLS components XS have maximum covariance. YS is an n-by-ncomp matrix with rows corresponding to observations and columns to components. YS is neither orthogonal nor normalized.
- ***PCTVAR*** is a 2-by-ncomp matrix containing the percentage of variance explained by the model. The first row of PCTVAR contains the percentage of variance explained in X by each PLS component, and the second row contains the percentage of variance explained in Y. (These values are R^2Y values.)
- ***MSE*** is a 2-by-(ncomp+1) matrix containing estimated mean-squared errors for PLS models with 0:ncomp components. The first row of MSE contains mean-squared errors for the predictor variables in X, and the second row contains mean-squared errors for the response variables in Y.
- ***Stats*** returns a structure with fields: W (a p-by-ncomp matrix of PLS weights, XS = X0*W), T2 (the T^2 statistic for each point in XS), Xresiduals (the predictor residuals, that is, X0 – XS*XL), and Yresiduals (the response residuals, that is, Y0 – XS*YL)

Displaying outputs:

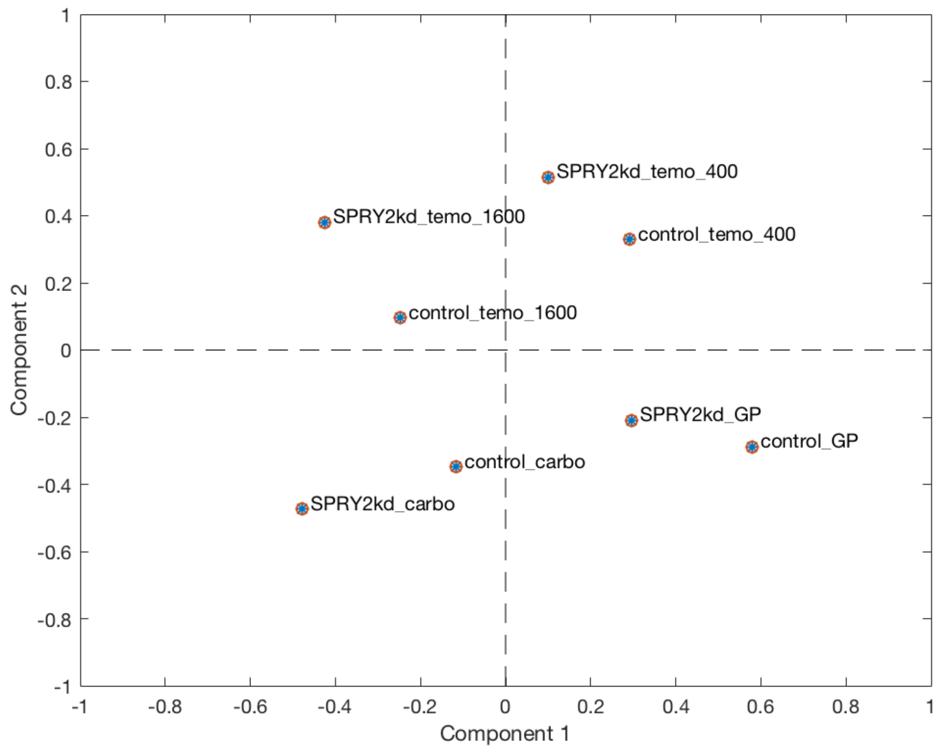
7. Follow the example code provided below to complete the percent variance in y explained by each component (R^2Y):

```
figure;
plot(1:ncomp,cumsum(100*pctvar(2,:)),'-bo');
xlabel('Components')
ylabel('Percent Variance Explained in Y')
```



8. Follow the example code below to create a scores plot (done below for first two components):

```
figure;
xlabel('Component 1')
ylabel('Component 2')
hold on
    x = x_scores(:,1);
    y = x_scores(:,2);
    scatter(x,y);
    xlim([-1 1]);
    ylim([-1 1]);
    box on
    hax = gca;
    line([0 0],get(hax,'YLim'),'Color','k','LineStyle','--')
    hline = refline([0 0]); hline.Color = 'k'; hline.LineStyle = '--';
    labels = X_row_labels;
    yline = 0;
    xline = 0;
    dx = 0.02; dy = 0.02; % displacement so the text does not overlay the data points
    text(x+dx, y+dy, labels, 'Fontsize', 10, 'Interpreter', 'none'); % labeling points
```



9. Follow the example code provided below to create a loadings plot:

This portion deals with duplicate values in the data before labeling. It is not necessary if there are none.

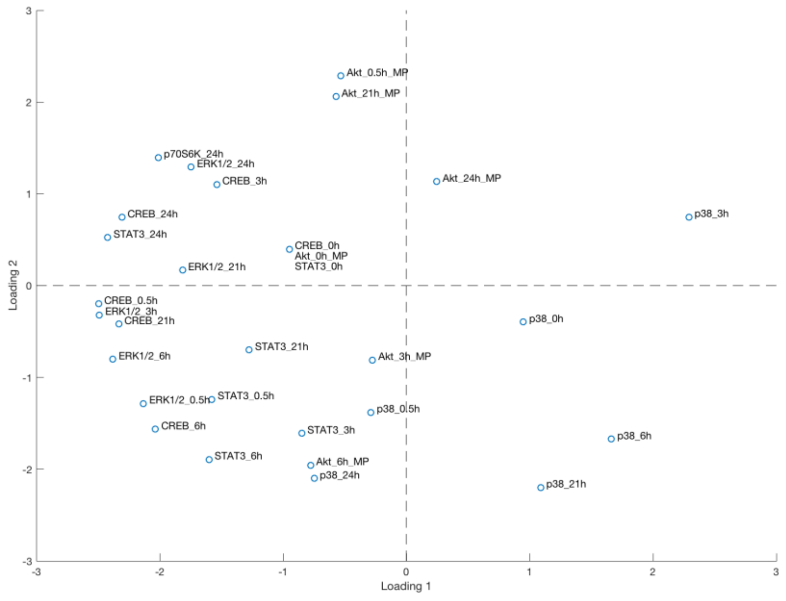
```
matches = zeros(max(size(x_loadings(:,2))),max(size(x_loadings(:,2)))); % allocate matrix
if size(unique(x_loadings(:,1:2))) ~= size(x_loadings(:,1:2)) % check if there all values are unique
   [C,ia,ic] = unique(x_loadings(:,1:2),'rows'); % getting all unique values
end
duplicate_ind = setdiff(1:size(x_loadings, 1), ia); % get indices of values that were repeated
duplicate_value = x_loadings(duplicate_ind, 1:2); % storing duplicate values

%% Plotting loading values
figure;
x1 = C(:,1); %if portion above is not used, replace C with x_loadings
y1 = C(:,2);
x_dup = x_loadings(duplicate_ind,1);
y_dup = x_loadings(duplicate_ind,2);
scatter(x1,y1,'o')
xlabel('Loading 1')
ylabel('Loading 2')
xlim([-3 3]); ylim([-3 3]); %adjust these limits as necessary to include all data points on your plot
box on
hax = gca;
line([0 0],get(hax,'YLim'),'Color','k','LineStyle','--')
hline = refline([0 0]); hline.Color = 'k'; hline.LineStyle = '--';
cell_col = X_col_labels(ia); %if portion above is not used remove ia and just leave X_row_labels
```

```
yline = 0; xline = 0;
dx = max(x_loadings(:,1))/50;
dy = max(x_loadings(:,2))/50; % displacement so the text does not overlay the points
text(x1+dx, y1+dy, cell_col, 'Fontsize', 10, 'Interpreter', 'none','Color','k');
```

The following portion shifts the labels on points with the same value. If there are no duplicate values this portion is again not necessary.

```
for i = 1:size(duplicate_value,1) - 1
    text(x_dup(i+1)+dx, y_dup(i+1)+dy-2.5*i*dy, X_col_labels(duplicate_ind(i+1)), 'Fontsize', 10, 'Interpreter',
'none','Color','k')
end
```



Cross-validation

10. To perform a PLSR calculation with cross-validation, modify the *plsregress* command accordingly:

```
ncomp = 5; %use only 5 components → change this to desired number of components
fcv = 5; %perform PLSR with 5-fold cross-validation -- this is an integer value, up to the total number of
        %rows in the input data sets (for this sample data set, can be up to 8)
mcreps = 1; %number of Monte-Carlo repetitions used during cross-validation

[x_loadings, y_loadings, x_scores, y_scores, beta, pctvar, mse, stats] = plsregress(z_x, z_y, ncomp, 'CV',
ncv,’mcreps’,mcreps);

%Calculate Q2Y values for each model component using mean squared errors from cross-validation. For %this particular
data set, the Q2Y values will be negative, indicating that the PLSR model developed here %is not a predictive one.
Q2Y = 1- mse(2,2:end)/sum(sum((z_y-mean(z_y)).^2)./size(z_y,1));
```