Partial Least Squares

Colin Adams, Alex Mun, Jackson Ray, Maleehah Zafar and Benjamin Zeitlin,

Team 4 - The Greatest Least Squares

11/29/2022

Section 1

Partial Least Squares (PLS) Introduction

Partial Least Squares (PLS) - Introduction

As the name indicates, **Partial Least Squares** is an alternative method to the Ordinary Least Squares and Principal Component Analysis. We commonly use OLS for linear regression when it comes to predict the response variable from multiple explanatory variables, given that the assumptions of linear regressions meet.

Why not just use OLS?

Multicollinearity is frequently seen in the datasets, which violates the said assumptions and disrupts the model's accuracy.

How can PLS prevent this?

Partial Least Squares counters multicollinearity by reducing the dimensionality of the correlated variables.

Partial Least Squares (PLS) - Introduction

- Partial Least Squares counters multicollinearity by reducing the dimensionality of the correlated variables.
- PLS uses a mix of PCA and Ordinary Least Squares to achieve this.
- Multicollinearity is when multiple predictor variables are somewhat linearly related / strongly correlated.

Quick Runthrough of PCA

Principal Component Analysis is used to mainly reduce the dimensionality of our given variables, while allowing us to preserve our correlation and variance between the set of correlated variables.

PCA will break up the given predictor variables into principle components that each correspond to the variability in the data. PC1 will always have the maximum variability, PC2 will have the second most variability, and so on.

It allows us to examine the variability between each principle component in the system.

Usage of PCA

- Principal component analysis is another key part of partial least squares.
- PCA is used over PLS when you need unsupervised learning, otherwise, PLS is a great supervised learning technique.
- PCA achieves dimension reduction meaning that we can avoid regression issues such as multicollinearity. PCA also guarantees that the data will be orthogonal.

PCA as a Dimension-Reduction Tool

What is dimension reduction?

Dimension reduction is the process of reducing the number of random variables. This is performed by transforming the data from a high-dimensional space space into a low-dimensional space.

- The dimension reduced data can be used for:
  - Cleaning the data
  - Visualizing and understanding the data
  - Building a simpler model

Why use PCA for dimension reduction?

- PCA helps capture highly correlated variables or dimensions.
- Orthogonal projection of the data onto a lower dimensional subspace.
- Variance of the projected data is maximized.

Dimension Reduction - A Manufacturing Example

Often in manufacturing, engineers will use dimension reduction to transform high-dimensional signals into low-dimensional features. These features are then monitored for process control (i.e. quality control charts).

- Example:
  - An automotive manufacturer wants to have the run rates of their machines monitored every 6 hours. However, the statistical process engineer (SPE) responsible for monitoring the machine would like to know the continuous metrics between these 6 hour intervals to ensure that the process is not out-of-control. Dimension reduction can be used to automatically trigger a flag or signal for significant changes in the process. The PCs are used to trigger an alarm once they surpass an established threshold.

Why would you use PLS over PCA and OLS? (aka. What makes PLS unique?)

OLS doesn't work well when the predictor variables are highly correlated with each other, meaning it can't handle multicollinearity.

Additionally, the accuracy of MLR decreases as the number of predictor variables increases. Thus, it is better to use PLS regression because it decreases the number of predictor variables beforehand.

That means use OLS when your data have small number of factors and they are not collinear.

PCA handles the variables by removing insignificant variables, but it does little to explain what they do in terms of predicting the response variable.

Still, it is unwise to assume that PLS is the solution for every situation.

Pros and Cons of PLS

Pros:

- Particularly useful when the matrix has more variables than observations.
- Takes into account Multicollinearity within the X matrix.
- Additionally, we can also include multiple correlated independent variables.
- Many variations of PLS for different scenarios.

Cons:

- It is prone to error when there are more observations than variables in the X matrix.
- OLS should be used if no multicollinearity.

Comparing the usage between PLS and OLS Example (1/2)

```
# Split into training and testing sets
set.seed(1)
train <- mtcars[1:25, c("hp", "mpg", "disp", "drat", "wt", "qsec")]
y_test <- mtcars[26:nrow(mtcars), c("hp")]
test <- mtcars[26:nrow(mtcars), c("mpg", "disp", "drat", "wt", "qsec")]
```

For an example, we can look at the usage between ols and pls on the mtcars dataset to predict hp.

Comparing the usage between PLS and OLS Example (2/2)

```
# Prediction for PLS
model <- plsr(hp ~ mpg + disp + drat + wt + qsec, data = train,
    scale = TRUE, validation = "CV")
pls_pred <- predict(model, test, ncomp = 2)

# RMSE of PLS Prediction
sqrt(mean((pls_pred - y_test)^2))
```

```
## [1] 54.89609
```

```
# Prediction for OLS
ols_model = lm(hp ~ mpg + disp + drat + wt + qsec, data=train)
ols_pred = predict(ols_model, test)

# RMSE of OLS Prediction
sqrt(mean((ols_pred - y_test)^2))
```

```
## [1] 57.94204
```

The RMSE for PLS is slightly lower than that of OLS. This is because PLS helps us determine the optimal number of principal components.

Section 2

Full Partial Least Squares Example

Validation and Training tables

Validation

The validation table of the model shows the test root mean squared error (test RMSE) that is calculated by the k-fold cross validation for each number of PLS components, or predictor variables. The interpretation of the change in this value with the addition of PLS components is that a lower RMSE value indicates a more optimal model at that number of PLS components.

Training

The training table shows the percentage of the response variable's variance that is explained by the components for each number of components. Although this percentage will always increase as you add more components, the number of components should be decided based upon the amount in which the percentage of the variance of the response variable increases, and whether or not this increase is significant.

Validation Plots

Validation Plots

These plots show the RMSE, MSE, and R-squared values for each number of components. Note that, for the RMSE and MSE validation plots, the best number of components is the number with the lowest value for RMSE or MSE, while the validation plot with R-squared values shows the best number of components with the highest value for R-squared.

Partial Least Squares on mtcars

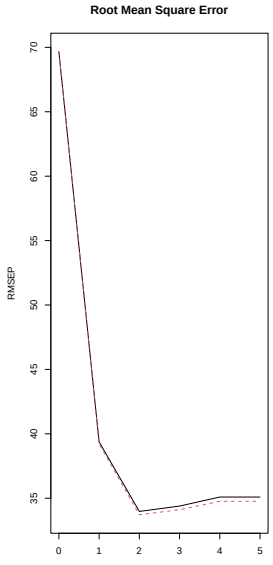
```
# Make example reproducible
set.seed(1)

# Randomize rows of the dataset
mtcars_data <- mtcars[sample(1:nrow(mtcars)),]

# Fit model
model <- plsr(hp ~ mpg + disp + drat + wt + qsec,
              data=mtcars_data, scale=TRUE, validation="CV")
```

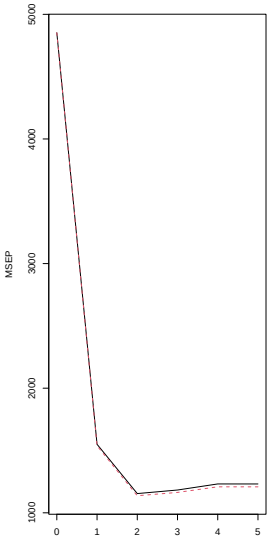
Scale=TRUE causes each of the variables in the dataset to be scaled to have a mean of 0 and a standard deviation of 1. Validation=CV causes function to use k-fold cross-validation to evaluate the performance of the model

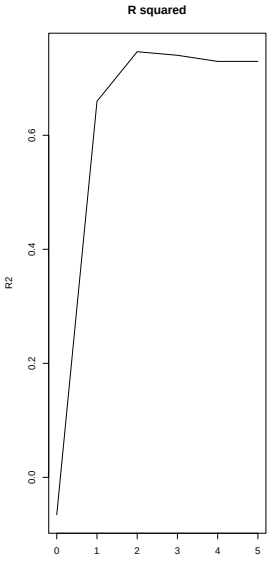
PLS Summary

```
Data:   X dimension: 32 5
    Y dimension: 32 1
Fit method: kernelpls
Number of components considered: 5

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps
CV           69.66    39.37    33.98    34.39    35.09    35.09
adjCV        69.66    39.22    33.72    34.11    34.76    34.76

TRAINING: % variance explained
    1 comps  2 comps  3 comps  4 comps  5 comps
X     68.66    89.27    95.82    97.94   100.00
hp    71.84    81.74    82.00    82.02    82.03
```

PLS Example Continued



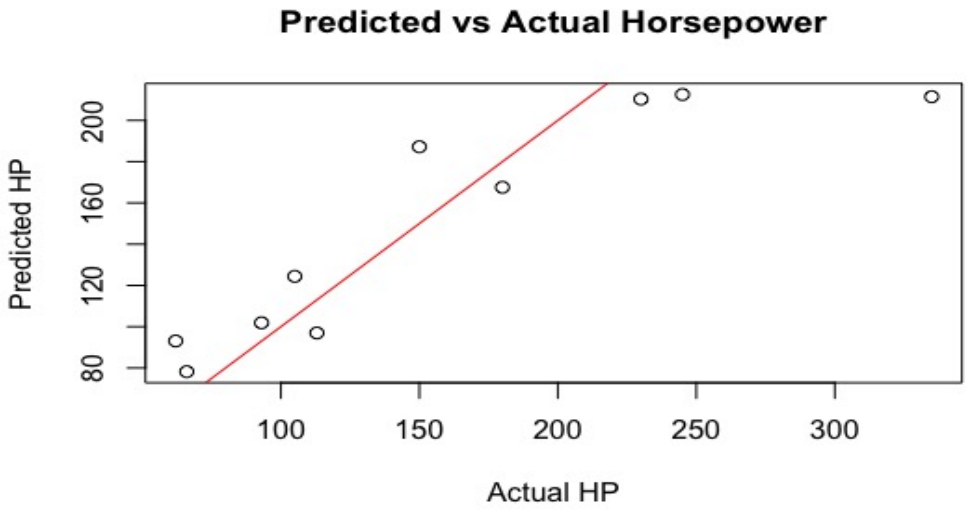




PLS Example Continued: Model Prediction

```
# Split into training and testing sets
train <- mtcars_data[1:22, c("hp", "mpg", "disp", "drat", "wt", "qsec")]
y_test <- mtcars_data[23:nrow(mtcars_data), c("hp")]
test <- mtcars_data[23:nrow(mtcars_data),
                    c("mpg", "disp", "drat", "wt", "qsec")]

# Use model to make predictions on a test set
model <- plsr(hp ~ mpg + disp + drat + wt + qsec,
              data=train, scale=TRUE, validation="CV")
pcr_pred <- predict(model, test, ncomp=2)
# Predicted vs Actual values
pred_vs_act <- as.data.frame(pcr_pred)
pred_vs_act$act <- y_test
colnames(pred_vs_act) <- c("Predicted HP", "Actual HP")
```

Predicted Versus Actual Graph



PLS Scoring

	Predicted HP	Actual HP
Merc 450SL	167.55433	180
Camaro Z28	212.54084	245
Lotus Europa	96.97024	113
Merc 240D	93.09389	62
Valiant	124.40128	105
Dodge Challenger	187.22092	150
Maserati Bora	211.47966	335
Datsun 710	101.87940	93
Fiat X1-9	78.22116	66
Chrysler Imperial	210.33478	230

Section 3

Partial Least Squares Theory

Theory behind PLS (1/3)

While multivariate multiple regression can account for multiple independent variables, The main reason in which we would use partial least squares vs Multivariate Multiple regression is to account for multicollinearity.

You would use Multivariate Multiple Regression only when you have a small amount of factors and non-collinear data.

We have seen the following equation many times

$$\boldsymbol{Y} = \boldsymbol{X}\beta + \boldsymbol{e}$$

Where $\boldsymbol{Y}$ is a $n \times 1$ vector for the predictor variables, $\boldsymbol{X}$ is a $n \times m$ matrix of the response variables, β are the (unknown) regression coefficients, and $\boldsymbol{e}$ is the residual error.

Theory behind PLS (2/3)

In order to find the unknown regression coefficients that will minimize the sum of squared errors of the residual error, we can do the following:

$$\hat{\beta} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}\boldsymbol{Y}$$

There is a problem that occurs with $(\boldsymbol{X}'\boldsymbol{X})^{-1}$ where it can often become singular for two reasons:

1. The number of columns exceeds the number of rows
2. There is collinearity between the columns

These issues can cause $\hat{\beta}$ to become ill-conditioned.

Theory behind PLS (3/3)

The second reason where there is a collinearity issue can be resolved with using either PCR or PLS and both of these use a similar approach to solving this issue.

In PCR, $\boldsymbol{X}$ is decomposed using singular value decomposition (SVD) and the following equation is formed:

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}' = \boldsymbol{T}\boldsymbol{P}'$$

Note that $\boldsymbol{X}$ is approximated by the low rank approximation by looking k largest singular values, so it is actually $\boldsymbol{X} \approx \boldsymbol{T}_k\boldsymbol{P}'_k$.

And we can find the regression coefficients by plugging in $\boldsymbol{T}\boldsymbol{P}'$ into $\hat{\beta} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}\boldsymbol{Y}$ to get the following:

$$\hat{\beta} = \boldsymbol{P}(\boldsymbol{T}'\boldsymbol{T})^{-1}\boldsymbol{T}'\boldsymbol{Y}$$

Where $\boldsymbol{T} = \boldsymbol{U}\boldsymbol{\Sigma}$ is a matrix of orthogonal scores and $\boldsymbol{P} = \boldsymbol{V}$ is a matrix of orthogonal loadings. Although this helps with collinearity issues, this only takes into account information about $\boldsymbol{X}$ since the left singular vectors of $\boldsymbol{X}$ multiplied with the singular values form the scores and the right singular values of $\boldsymbol{X}$ form the loadings.

Methodology

Step 1: Construct the SVD of the matrix $\boldsymbol{S} = \boldsymbol{X}'\boldsymbol{Y}$

Step 2: Use the first left singular vector which we'll call w and the first right singular vector called q to construct scores t and u (Note that in the first iteration $\boldsymbol{E} = \boldsymbol{X}$ and $\boldsymbol{F} = \boldsymbol{Y}$)

$$\begin{aligned} t &= \boldsymbol{E}w \\ u &= \boldsymbol{F}q \end{aligned}$$

Step 3: Normalize the t scores

Methodology Continued

Step 4: Construct the X and Y loadings by doing the following:

$$\begin{aligned} p &= \boldsymbol{E}'t \\ q &= \boldsymbol{F}'t \end{aligned}$$

Step 5: Update $\boldsymbol{E}$ and $\boldsymbol{F}$

$$\begin{aligned} \boldsymbol{E}_{n+1} &= \boldsymbol{E}_n - tp' \\ \boldsymbol{F}_{n+1} &= \boldsymbol{F}_n - tq' \end{aligned}$$

Step 6: Repeat the process

During the iterations, the vectors t, u, p, and q will be saved into matrices $\boldsymbol{T}$, $\boldsymbol{U}$, $\boldsymbol{P}$, and $\boldsymbol{Q}$.

Now that we have matrices $\boldsymbol{T}$ and $\boldsymbol{P}$, we can do the same as what was done in PCR where $\boldsymbol{X} = \boldsymbol{T}\boldsymbol{P}'$ and $\hat{\beta} = \boldsymbol{P}(\boldsymbol{T}'\boldsymbol{T})^{-1}\boldsymbol{T}'\boldsymbol{Y}$. In PLS as well, only the first a components are used which is determined by cross-validation.

Section 4

Partial Least Squares Regression Example using Global Emissions

PLS Big Data Dictionary

Table 1: Fundamental Data Values Used

Variable Name	Variable Type	Description
Country.GDP	Integer Value	A measure of Economic Performance by Country.
Emissions.Production.CO2	Integer Value	CO2 total emissions production.
Emissions.Production.CH4	Integer Value	CH4 Emissions production.
Emissions.Production.N2O	Integer Value	N2O Emissions production per country.
Year	Integer Value	The measure of data for the Year's worth.

The data set we used describes the yearly output of harming emissions produced by each country. This includes toxic chemicals such as Nitrous Oxide, Methane, Extra Flaring, Carbon Dioxide Emissions by fossil fuels, etc.

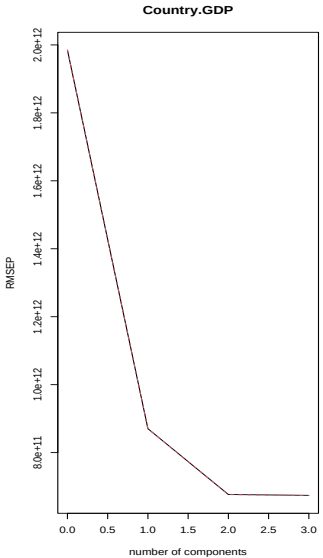
Readying our Data Set

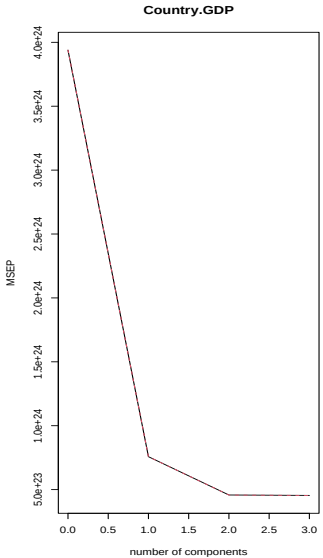
```
#make example reproducible
set.seed(1)

emiss_data <- emiss_dat[sample(1:nrow(emiss_dat)), ]

# Fit PCR model
# Scale=TRUE causes each of the variables in the dataset to be scaled to
# have a mean of 0 and a standard deviation of 1
# Validation=CV causes function to use k-fold cross-validation to
# evaluate the performance of the model
mod_emiss <- plsr(Country.GDP ~ Emissions.Production.CO2.Total +
                    Emissions.Production.CH4 + Emissions.Production.N2O,
                  data=emiss_data, scale=TRUE, validation="CV")
```

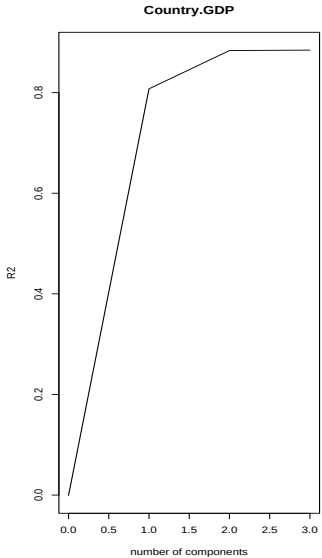
Global Emissions Fitted Summary

```
Data:   X dimension: 2484 3
    Y dimension: 2484 1
Fit method: kernelpls
Number of components considered: 3

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)    1 comps    2 comps    3 comps
CV       1.985e+12  8.703e+11  6.766e+11  6.742e+11
adjCV    1.985e+12  8.703e+11  6.763e+11  6.738e+11

TRAINING: % variance explained
             1 comps  2 comps  3 comps
X              92.05    97.87   100.00
Country.GDP    80.88    88.58    88.68
```

Cross Validation Plots







Splitting the training and test data

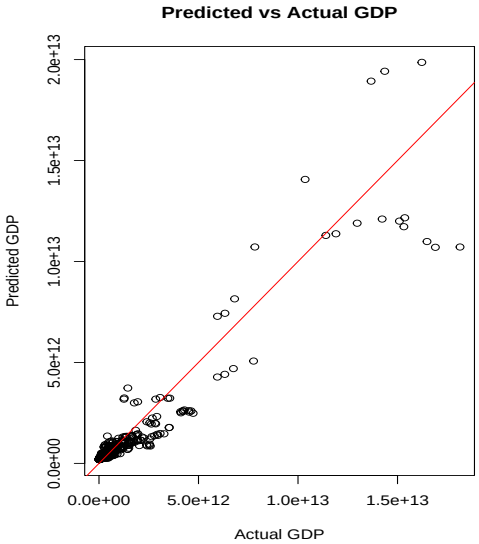
```
# Split into training and testing sets
train <- emiss_data[1:1860,c("Country.GDP","Emissions.Production.CO2.Total",
        "Emissions.Production.CH4","Emissions.Production.N2O")]

y_test <- emiss_data[c(1861:nrow(emiss_data)), c("Country.GDP")]
test <- emiss_data[c(1861:nrow(emiss_data)),
        c("Emissions.Production.CO2.Total", "Emissions.Production.CH4",
          "Emissions.Production.N2O")]

# Use model to make predictions on a test set
mod_emiss <- plsr(Country.GDP ~ Emissions.Production.CO2.Total +
            Emissions.Production.CH4 + Emissions.Production.N2O, data=train,
            scale=TRUE, validation="CV")
pcr_pred <- predict(mod_emiss, test, ncomp=3)
```

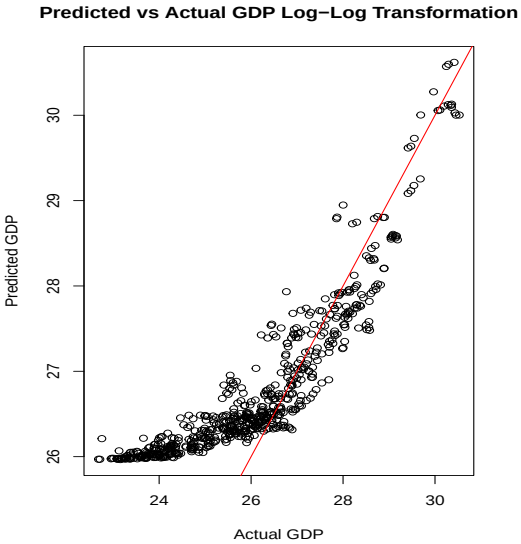
Checking for Accuracy and Plotting the Data

Using the plots, we can evaluate the accuracy of our fits and analyze the graphs further.

```
par(mfrow = c(1,2))
plot(y_test, pcr_pred, main = "Predicted vs Actual GDP",xlab = "Actual GDP",
     ylab = "Predicted GDP")
abline(0, 1, col='red')
plot(log(y_test), log(pcr_pred),
     main = "Predicted vs Actual GDP Log-Log Transformation",
     xlab = "Actual GDP", ylab = "Predicted GDP")
abline(0, 1, col='red')
```

Plotting the Predicted vs Actual GDP





Partial Least Squares vs Multivariate Multiple Regression

While multivariate multiple regression can account for multiple independent variables, The main reason in which we would use partial least squares vs Multivariate Multiple regression is to account for multicollinearity. You would use Multivariate Multiple Regression only when you have a small amount of factors and non-collinear data. For this, we can plot with line made from MLR, PLS, and actual data to show how off the MLR is compared to PLS

PLS1 and PLS2

When it comes to achieving a response variable, we have the choice between returning multiple responses or a singular response.

PLS1 is used when we are returning a single response, while PLS2 is used when we are returning multiple responses. *Note: The response is not limited to single type of data

PLSDA

PLSDA is a Supervised Learning Classification Method branching off from PLS classification. This is generally used when we want to supervise the data. We'll be classifying cars based on the number of gears each car has. Using the dataset - mtcars:

```
data(mtcars)

# split the data randomly 50%
index <- sample(1:nrow(mtcars), round(nrow(mtcars) * 0.5))

# X matrices
Xc <- mtcars[index, 1:7]
Xv <- mtcars[-index, 1:7]

# Y matrices
cc.all <- as.factor(mtcars[index, 10])
cv.all <- as.factor(mtcars[-index, 10])
```

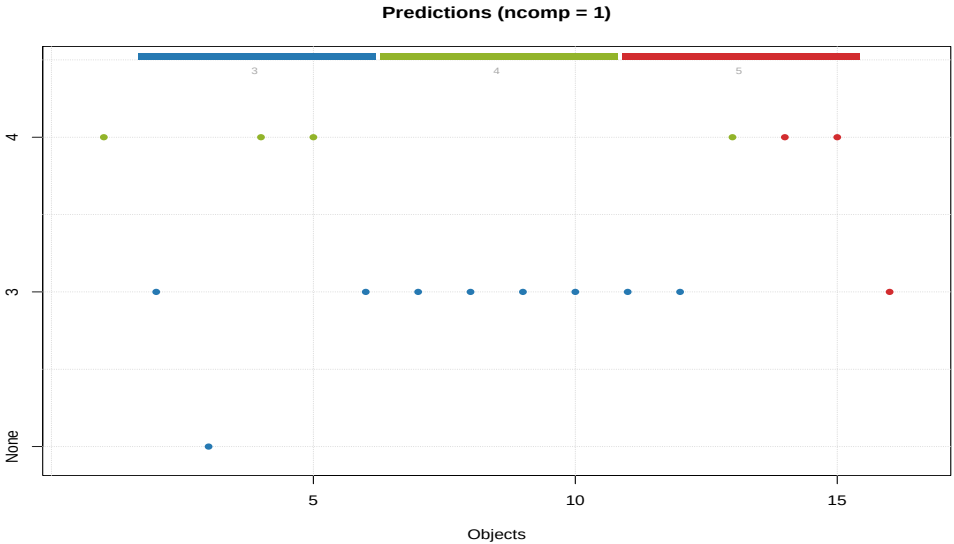
PLSDA Summary

We can use the summary function to evaluate the accuracy of each class. In this example, we showed the first class.

```
PLS-DA model (class plsda) summary
-----------------------------------
Info:
Number of selected components: 1
Cross-validation: full (leave one out)

Class #1 (3)
    X cumexpvar Y cumexpvar TP FP TN FN Spec. Sens. Accuracy
Cal          94       37.35  4  1  9  2   0.9 0.667    0.812
Cv           NA          NA  4  1  9  2   0.9 0.667    0.812
```

Freq

PLSDA Predictions Plot



PLSDA Confusion Matrix

	3	4	5	None
3	8	0	0	1
4	0	4	0	0
5	1	2	0	0

By examining the diagonal, we can determine the that the true values predicted 11 of our 14 rows correctly while having 3 incorrect values. The accuracy on this would be 78.57%.

Non-Linear Cases (Kernel PLS)

When using PLS, we are assuming linearity within our data. However, this is not always the case. In circumstances where we do, we use kernels to apply PLS for nonlinear cases.

To kernelize PLS, we have two options:

1. Similar to methodology used in SVM, map each point nonlinearly in high-dimensional space, construct linear regression in mapped space corresponding to nonlinear functions in the original space, and replace the mapped data that appears as dot product with kernel functions.
2. Use PLS to factorize the kernel matrix directly.

References

Click on the categories to open the links.

PLS Theory:

PLS Theory Ref1
PLS Theory Ref2

PLS:

PLS Regression Ref1 General PLS Ref2 PLS Latent Design Ref3

PCA:

PCA

References Continued

Global Emissions Data:

Emissions Data Ref1

mtcars data set:

MTcars

Kernel PLS Ref

Kernel PLS Ref1 Kernel PLS Ref2

PLS/PCA for Manufacturing

Manufacturing Example