8

PROJECTION METHODS IN EXCEL*

This chapter describes the Chemometrics Add-In, which is an Add-In designed especially for Microsoft Excel. The Chemometrics Add-In enables data analysis on the basis of the following projection methods: Principal Component Analysis (PCA) and Projection on Latent Structures (PLS1 and PLS2).

Preceding chapters	2, 7
Dependent chapters	9–13, 15
Matrix skills	Basics
Statistical skills	Low
Excel skills	Basic
Chemometric skills	Basic
Chemometrics Add-In	Used
Accompanying workbook	Projection.xls

8.1 PROJECTION METHODS

8.1.1 Concept and Notation

Projection methods are widely used for multivariate data analysis, especially in chemometrics. They are applied both to one-block data **X** (classification, e.g., PCA) and to double-block data such as **X** and **Y** (calibration, e.g., principal component regression (PCR) and partial least square (PLS)).

*With contributions from Oxana Rodionova.

Chemometrics in Excel, First Edition. Alexey L. Pomerantsev.

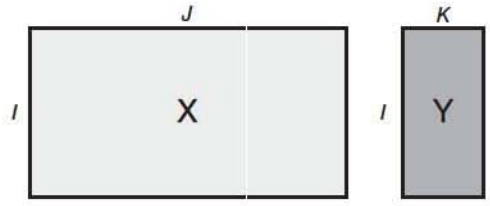


Figure 8.1 Multivariate data.

Let us consider a $(I \times J)$ data matrix $\mathbf{X}$, where I is the number of objects (rows) and J is the number of independent variables (columns). Ordinarily, the number of variables is rather high ($J \gg 1$). We can simultaneously analyze the $(I \times K)$ matrix $\mathbf{Y}$, where I is the same number of objects and K is the number of responses. See Fig. 8.1.

The essence of the projection techniques is a considerable reduction of data dimensionality for both blocks $\mathbf{X}$ and $\mathbf{Y}$. There are many reviews on projection methods, and the reader is invited to refer to them for more detail.

8.1.2 PCA

PCA is the oldest projection method. This method uses new formal (or latent) variables $\mathbf{t}_a (a = 1, \ldots, A)$, which are linear combinations of the original variables $\mathbf{x}_j (j = 1, \ldots, J)$, that is,

$$\mathbf{t}_a = \mathbf{p}_{a1}\mathbf{x}_1 + \cdots + \mathbf{p}_{aJ}\mathbf{x}_J,$$

or in the matrix notation

$$\mathbf{X} = \mathbf{T}\mathbf{P}^{\mathrm{t}} + \mathbf{E} = \sum_{a=1}^{A} \mathbf{t}_a \mathbf{p}_a^{\mathrm{t}} + \mathbf{E} \tag{8.1}$$

In this equation, $\mathbf{T}$ is called the *scores matrix* or *scores*. Its dimension is $(I \times A)$. Matrix $\mathbf{P}$ is called the *loadings matrix*, or *loadings*, and it has a dimension $(A \times J)$. $\mathbf{E}$ is the *residuals* $(I \times J)$ matrix. See Fig. 8.2.

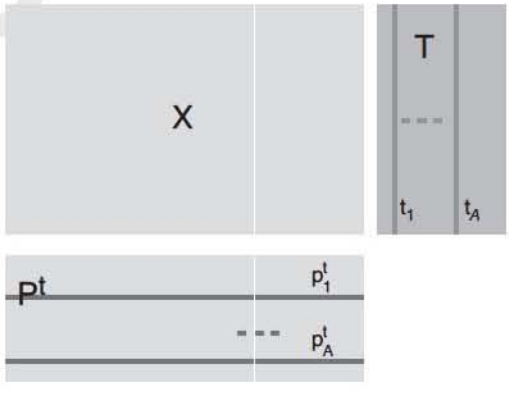


Figure 8.2 Graphic representation of PCA.

New variables $\mathbf{t}_a$ are often called *principal components* (*PCs*); therefore, the method is called *PCA*. The number of columns $\mathbf{t}_a$ in matrix $\mathbf{T}$ and columns $\mathbf{p}_a$ in matrix $\mathbf{P}$ is equal to A, which is the *number of PCs*. It defines the projection model complexity. The value of A is certainly less than the number of variables J and the number of objects I.

Scores and loadings matrices have the following properties

$$\mathbf{T}^t\mathbf{T} = \Lambda = \mathrm{diag}\{\lambda_1, \ldots, \lambda_A\}, \quad \mathbf{P}^t\mathbf{P} = \mathbf{I}$$

The recurrent algorithm, non-linear iterative partial least squares (NIPALS) (see Section 15.5.3) is often used for calculation of PCA scores and loadings. After constructing the PC space, new objects $\mathbf{X}_{new}$ can be projected onto this space. In other words, the scores matrix $\mathbf{T}_{new}$ can be calculated. In PCA, this is easily done by equation

$$\mathbf{T}_{new} = \mathbf{X}_{new}\mathbf{P}.$$

Naturally, $\mathbf{X}_{new}$ matrix should be preprocessed in the same way as the training matrix $\mathbf{X}$, which was used for PCA decomposition

8.1.3 PLS

PLS can be considered as a generalization of PCA. In PLS, the decomposition of matrices $\mathbf{X}$ and $\mathbf{Y}$ is conducted simultaneously

$$\mathbf{X} = \mathbf{T}\mathbf{P}^t + \mathbf{E}, \quad \mathbf{Y} = \mathbf{U}\mathbf{Q}^t + \mathbf{F}, \quad \mathbf{T} = \mathbf{X}\mathbf{W}\left(\mathbf{P}^t\mathbf{W}\right)^{-1} \tag{8.2}$$

A projection is built in order to maximize correlation between corresponding vectors of $\mathbf{X}$-scores $\mathbf{t}_a$ and $\mathbf{Y}$-scores $\mathbf{u}_a$. See Fig. 8.3 for illustration.

If block $\mathbf{Y}$ consists of several responses (i.e., $K > 1$), two types of projections can be built; they are PLS1 and PLS2. In the first case, the projection space is built separately for each response variable $\mathbf{y}_k$. Scores $\mathbf{T}$ ($\mathbf{U}$) and loadings $\mathbf{P}$ ($\mathbf{W}$, $\mathbf{Q}$) depend on the response $\mathbf{y}_k$

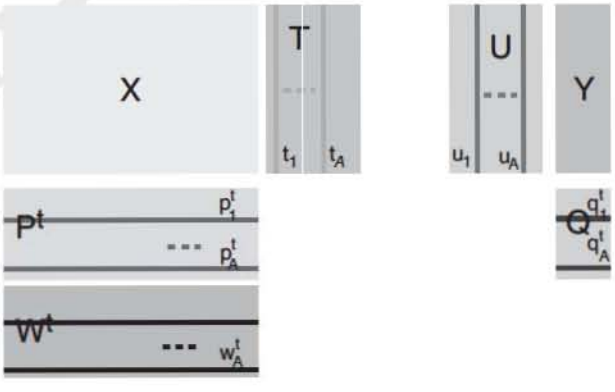


Figure 8.3 PLS2 graphic representation.

in use. Such an approach is called *PLS1*. In the *PLS2* method, one common space is built for all responses. The following expressions show the properties of PLS projection matrices

$$\mathbf{T}^t\mathbf{T} = \Lambda = \mathrm{diag}\{\lambda_1, \ldots, \lambda_A\}, \quad \mathbf{W}^t\mathbf{W} = \mathbf{I}, \quad \mathbf{T} = \mathbf{XR}, \quad \mathbf{R} = \mathbf{W}\left(\mathbf{P}^t\mathbf{W}\right)^{-1}$$
$$\mathbf{R}^t\mathbf{P} = \mathbf{I}, \quad \mathbf{Y}^{\mathrm{hat}} = \mathbf{XB} = \mathbf{TQ}^t, \quad \mathbf{B} = \mathbf{RQ}^t, \quad \mathbf{Q}^t = \Lambda^{-1}\mathbf{T}^t\mathbf{Y}, \quad \mathbf{W}^t = \Lambda^{-1}\mathbf{U}^t\mathbf{X}$$

A recurrent algorithm is used for the calculation of PLS scores and loadings. This algorithm calculates one PLS component at a time. It is presented in Section 15.5.4. A similar algorithm for PLS2 is described in Section 15.5.5.

8.1.4 Data Preprocessing

It is worth mentioning that in the course of **X** and **Y** decomposition, the PCA and PLS methods do not take into consideration the free term. This could be seen from Eqs 8.1 and 8.2. It is initially supposed that all columns in matrices **X** и **Y** have zero mean values, that is,

$$\sum_{i=1}^{I} x_{ij} = 0 \quad \text{and} \quad \sum_{i=1}^{I} y_{ik} = 0, \quad \text{for each } j = 1, \ldots, J \text{ and } k = 1, \ldots, K$$

This condition can be easily satisfied by data *centering*.

Centering implied a subtraction of matrix **M** from the original matrix **X**, that is,

$$\widetilde{\mathbf{X}} = \mathbf{X} - \mathbf{M}$$

Centering is a column-wise operation. For each vector $\mathbf{x}_j$, the mean value is calculated by

$$m_j = \left(x_{1j} + \cdots + x_{Ij}\right)/I.$$

In that case, $\mathbf{M} = (m_1\mathbf{1}, \ldots, m_J\mathbf{1})$, where **1** is the $(I \times 1)$ vector of ones. Centering is a compulsory procedure that precedes the application of projection methods. The second simplest preprocessing technique is scaling.

Scaling is not as indispensable as centering. Compared to centering, scaling does not change the data structure but simply modifies the weights of different parts of data. The most widely used scaling is a column-wise one. This can be expressed as a right multiplication of matrix **X** by matrix **W**, that is,

$$\widetilde{\mathbf{X}} = \mathbf{XW}$$

Matrix **W** is the $(J \times J)$ diagonal matrix. As a rule, the diagonal elements w_{jj} are equal to inverse values of the standard deviations, that is,

$$d_j = \sqrt{\sum_{i=1}^{I} \left(x_{ij} - m_j\right)^2 \Big/ I}$$

calculated for each column $\mathbf{x}_j$. Row-wise scaling (also called *normalization*) is a left multiplication of matrix **X** by a diagonal matrix **W**, that is,

$$\widetilde{\mathbf{X}} = \mathbf{WX}$$

In this formula, $\mathbf{W}$ is the $(I \times I)$ diagonal matrix, and its elements $\mathbf{w}_{ii}$ are ordinarily the inverse values of the standard deviations calculated for each row $\mathbf{x}_i^t$

Combination of centering and column-wise scaling

$$\tilde{x}_{ij} = (x_{ij} - m_j)/d_j$$

is called *autoscaling*.

Data scaling is often used to make the contribution of various variables to a model more equal (i.e., in hybrid methods like LC–MS), to account for heterogeneous errors, or when different data blocks should be combined in one model. Scaling can also be seen as a method for stabilization of numerical calculations. At the same time, scaling should be used with caution as such preprocessing can essentially change the results of quality analysis. When the raw data structure is a priori assumed homogeneous and homoscedastic, data preprocessing is not only unnecessary but also harmful. This is discussed in Chapter 12 in the example of HPLC–DAD data.

Every type of preprocessing (centering, scaling, etc.) is first applied to the calibration dataset. This set is used to calculate the values of m_j and d_j, which, in turn, are used for preprocessing of both training and test sets later on.

In Chemometrics Add-In, preprocessing is done automatically. If there is a need in any particular preprocessing, it can be done using standard worksheet functions, or special user-defined functions, which are explained in Sections 13.1.2 and 13.2.3.

8.1.5 Didactic Example

File Projection.xls is used to illustrate the Chemometrics Add-In facilities. The performance of all the above-mentioned methods is illustrated with a simulated dataset (**X**, **Y**), which should be centered but not scaled.

File Projection.xls includes the following worksheets:

Intro: short introduction

Data: data used in the example. Block **X** consists of 14 objects (9 in the calibration set and 5 in the test set) and 50 variables. Block **Y** includes two responses, which correspond to the 14 objects. The worksheet also presents a legend that explains the names of the arrays

PCA: application of the worksheet functions **ScoresPCA** and **LoadingsPCA**

PLS1: application of the worksheet functions **ScoresPLS**, **UScoresPLS**, **LoadingsPLS**, **WLoadingsPLS**, and **QLoadingsPLS**

PLS2: application of the worksheet functions **ScoresPLS2**, **UScoresPLS2**, **LoadingsPLS2**, **WLoadingsPLS2**, and **QLoadingsPLS2**

Plus: application of the additional worksheet functions **MIdent**, **MIdentD2**, **MTrace**, and **MCutRows**

Unscrambler: comparison of the results obtained by Chemometrics Add-In and by the Unscrambler program

SIMCA-P: comparison of the results obtained by Chemometrics Add-In and by the SIMCA program

8.2 APPLICATION OF CHEMOMETRICS ADD-IN

8.2.1 Installation

Before starting to work with Chemometrics Add-In, the software should be properly installed. The instructions for Chemometrics Add-In installation are presented in Chapter 3.

8.2.2 General

Chemometrics Add-In uses all input data from an active Excel workbook. This information should be pasted directly in the worksheet area (data **X** and **Y**). A user may organize the working space the way he/she likes, that is, to place all data onto one worksheet, or split them between several worksheets, or even use worksheets from different workbooks. The results are returned as the worksheet arrays. The software does not have any limitations on the size of input arrays, that is, the number of objects (I), the number of variables (J), and the number of responses (K). The dimension is only limited by the supported computer memory and limitations of Excel itself.

Chemometrics Add-In includes user-defined functions, which can be used as standard worksheet functions. For this purpose, a user can employ **Insert Function** dialog box and select the **User Defined** category. All functions, described below can be found in the **Select a function** window.

The returned values are arrays; therefore, the functions should be entered in array formula style, that is, applying **CTRL+SHIFT+ENTER** at the end of the formula input (see Section 7.2.1).

Function arguments can be numbers, names, arrays, or references containing numbers. To insert an array formula, it is necessary to select a range, the size of which corresponds to the expected output array. If a selected range is larger than the output array, the superfluous cells are filled with #N/A symbols. On the contrary, when the selected range is smaller than the output array, a part of the output information is lost.

Number of Principal/PLS Components

Each function has an optional argument `PC` that defines the number of PCs (A). If `PC` is omitted, the output corresponds to the selected area. If PC value is more than $min\ (I, J)$, the decomposition is done for the maximum possible number of PCs and superfluous cells are filled with #N/A symbols.

Centering and/or Scaling

Each function has optional arguments **`CentWeightX`** and **`CentWeightY`**, which define whether centering and/or scaling for **X** and **Y** arrays is performed. These arguments can be as follows:

0 – no centering and no scaling (default value)
1 – only centering, that is, subtraction of column-wise mean values
2 – only scaling by column-wise standard deviations
3 – centering and scaling, that is, autoscaling

If any argument **`CentWeightX`** or **`CentWeightY`** is omitted it is assumed to be (equal to) **`0`**.

8.3 PCA

8.3.1 `ScoresPCA`

`ScoresPCA` performs decomposition of the matrix **X** using the PCA method (Eq. 8.1) and then returns the array of score values $\mathbf{T}_{new}$ calculated for matrix $\mathbf{X}_{new}$.

Syntax

```
ScoresPCA (X [, PC] [,CentWeightX] [, Xnew])
```

X is the array of **X**-values (calibration set);

`PC` is an optional argument (integer), which defines the number of PCs (A), used in the PCA decomposition;

`CentWeightX` is an optional argument (integer) that indicates whether centering and/or scaling is done;

`Xnew` is an optional argument that presents an array of new values $\mathbf{X}_{new}$ (test set) for which the score values $\mathbf{T}_{new}$ are calculated.

Remarks

- The arrays `Xnew` and **X** must have the same number of columns;
- If argument `Xnew` is omitted, it is assumed to be the same as **X**, and thus the calibration score values **T** are returned;
- The result is an array (matrix) where the number of rows equals the number of rows in array `Xnew`, and the number of columns equals the number of PCs (A);
- **`TREND`** is a similar standard worksheet function (Section 7.2.7).

Example

An example is given in worksheet PCA and shown in Fig. 8.4.

`ScoresPCA` is an array function which must be completed by **CTRL+SHIFT+ENTER**.

8.3.2 `LoadingsPCA`

Performs decomposition of matrix **X** using the PCA method (Eq. 8.1) and returns an array of loading values **P**.

Syntax

```
LoadingsPCA (X [, PC] [, CentWeightX])
```

X is the array of **X**-values (calibration set);

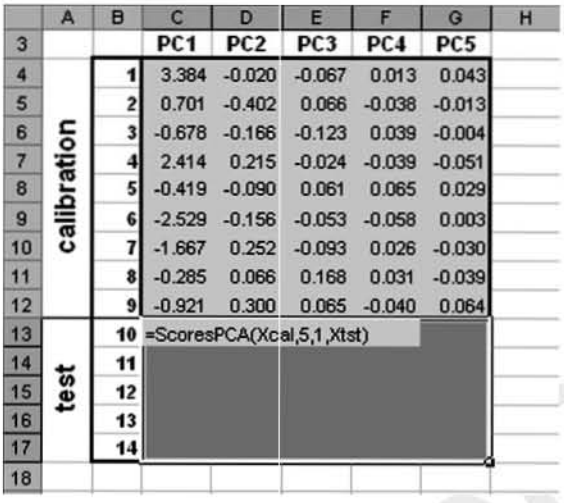


Figure 8.4 Example of **ScoresPCA** function.

`PC` is an optional argument (integer), which defines the number of PCs (A), used in PCA decomposition;

`CentWeightX` is an optional argument (integer) that indicates whether centering and/or scaling is done.

Remarks

- The result is an array (matrix) where the number of rows is equal to the number of columns (J) in array **X**, and the number of columns is equal to the number of PCs (A);
- **MINVERSE** is a similar standard worksheet function (Section 7.2.5).

Example

An example is given in worksheet PCA and shown in Fig. 8.5.

LoadingsPCA is an array function, which must be completed by **CTRL+SHIFT+ENTER**.

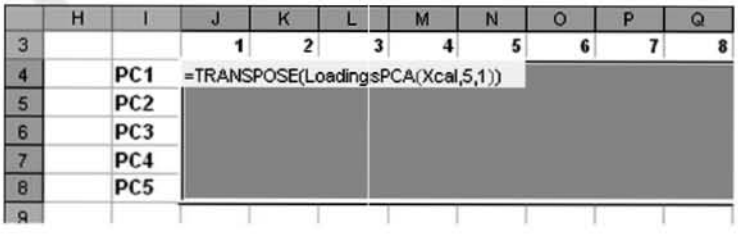


Figure 8.5 Example of **LoadingsPCA** function.

8.4 PLS

8.4.1 `ScoresPLS`

Performs decomposition of matrices **X** and **Y** using the method of PLS (Eq. 8.2) and then returns an array that presents the PLS score values $\mathbf{T}_{new}$ calculated for matrix $\mathbf{X}_{new}$.

Syntax

`ScoresPLS` `(X, Y [, PC] [, CentWeightX] [, CentWeightY] [, Xnew] )`

X is the array of **X**-values (calibration set);

Y is the array of **Y**-values (calibration set);

`PC` is an optional argument (integer), which defines the number of PLS components (*A*), used in PLS decomposition;

`CentWeightX` is an optional argument (integer) that indicates whether centering and/or scaling for matrix **X** is done;

`CentWeightY` is an optional argument (integer) that indicates whether centering and/or scaling for matrix **Y** is done;

`Xnew` is an optional argument that presents an array of new values $\mathbf{X}_{new}$ (test set) for which the PLS score values $\mathbf{T}_{new}$ are calculated.

Example

An example is given in worksheet PLS1 and shown in Fig. 8.6.

	A	B	C	D	E	F	G	H
3			PC1	PC2	PC3	PC4	PC5	
4	calibration	1	3.364	0.026	0.065	-0.016	-0.023	
5		2	0.742	0.384	-0.085	-0.042	-0.019	
6		3	-0.652	0.189	0.126	0.057	0.023	
7		4	2.373	-0.213	0.030	0.000	0.045	
8		5	-0.406	0.079	-0.058	0.048	-0.021	
9		6	-2.492	0.167	0.043	-0.043	0.037	
10		7	-1.683	-0.230	0.101	-0.008	-0.058	
11		8	-0.294	-0.095	-0.161	0.041	0.003	
12		9	-0.951	-0.308	-0.061	-0.037	0.013	
13	test	10	=ScoresPLS(Xcal,YcalA,5,1,1,Xtst)					
14		11						
15		12						
16		13						
17		14						
18								

Figure 8.6 Example of **`ScoresPLS`** function.

Remarks

- Array `Y` must have only one column ($K = 1$);
- Arrays `Y` and `X` must have the same number of rows (I);
- Arrays `Xnew` and `X` must have the same number of columns (J);
- If argument `Xnew` is omitted, it is assumed to be the same as `X` and thus the PLS calibration score values **T** are returned;
- The result is an array (matrix) where the number of rows equals the number of rows in array `Xnew`, and the number of columns equals the number of PCs (A);
- **TREND** is a similar standard worksheet function (Section 7.2.7).

ScoresPLS is an array function, which must be completed by **CTRL+SHIFT+ENTER**.

8.4.2 UScoresPLS

Performs decomposition of matrices **X** and **Y** using the method of PLS (Eq. 8.2) and then returns an array that presents the PLS score values $\mathbf{U}_{new}$ calculated for matrices $\mathbf{X}_{new}$ and $\mathbf{Y}_{new}$.

Syntax

```
UScoresPLS (X, Y, [, PC] [, CentWeightX] [, CentWeightY]
[, Xnew] [, Ynew])
```

`X` is the array of **X**-values (calibration set);

`Y` is the array of **Y**-values (calibration set);

`PC` is an optional argument (integer), which defines the number of PLS components (A), used in PLS decomposition;

`CentWeightX` is an optional argument (integer) that indicates whether centering and/or scaling for matrix **X** is done;

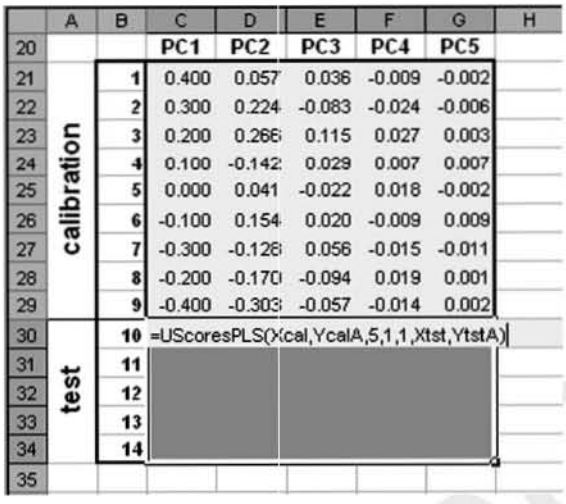
`CentWeightY` is an optional argument (integer) that indicates whether centering and/or scaling for matrix **Y** is done;

`Xnew` is an optional argument that presents an array of new values $\mathbf{X}_{new}$ (test set) for which PLS score values $\mathbf{U}_{new}$ are calculated;

`Ynew` is an optional argument that presents an array of new values $\mathbf{Y}_{new}$ (test set) for which PLS score values $\mathbf{U}_{new}$ are calculated.

Remarks

- Arrays `Y` and `Ynew` must have only one column ($K = 1$);
- Arrays `Y` and `X` must have the same number of rows (I);
- Arrays `Xnew` and `X` must have the same number of columns (J);



	A	B	C	D	E	F	G	H
20			PC1	PC2	PC3	PC4	PC5	
21	calibration	1	0.400	0.057	0.036	-0.009	-0.002	
22		2	0.300	0.224	-0.083	-0.024	-0.006	
23		3	0.200	0.266	0.115	0.027	0.003	
24		4	0.100	-0.142	0.029	0.007	0.007	
25		5	0.000	0.041	-0.022	0.018	-0.002	
26		6	-0.100	0.154	0.020	-0.009	0.009	
27		7	-0.300	-0.128	0.056	-0.015	-0.011	
28		8	-0.200	-0.170	-0.094	0.019	0.001	
29		9	-0.400	-0.303	-0.057	-0.014	0.002	
30	test	10	=UScoresPLS(Xcal,YcalA,5,1,1,Xtst,YtstA)					
31		11						
32		12						
33		13						
34		14						
35								

Figure 8.7 Example of **UScoresPLS** function.

- If argument Xnew is omitted, it is assumed to be the same as **X** and thus the PLS calibration score values **U** are returned;
- If argument Ynew is omitted, it is assumed to be the same as **Y**, and thus the PLS calibration score values **U** are returned;
- The result is an array (matrix) where the number of rows equals the number of rows in array Xnew, and the number of columns equals the number of PCs (A);
- **TREND** is a similar standard worksheet function (Section 7.2.7).

Example

An example is given in worksheet PLS1 and shown in Fig. 8.7.

UScoresPLS is an array function, which must be completed by **CTRL+SHIFT+ENTER**.

8.4.3 LoadingsPLS

Performs decomposition of matrices **X** and **Y** using the method of PLS (Eq. 8.2) and then returns an array that presents the loadings values **P**.

Syntax

LoadingsPLS (**X**, **Y** [, PC] [, CentWeightX] [, CentWeightY])

X is the array of **X**-values (calibration set);

Y is the array of **Y**-values (calibration set);

`PC` is an optional argument (integer), which defines the number of PLS components (A), used in the PLS decomposition;

`CentWeightX` is an optional argument (integer) that indicates whether centering and/or scaling for matrix **X** is done;

`CentWeightY` is an optional argument (integer) that indicates whether centering and/or scaling for matrix **Y** is done.

Remarks

- Array `Y` must have only one column ($K = 1$);
- Arrays `Y` and `X` must have the same number of rows (I);
- The result is an array (matrix) where the number of rows equals the number of columns (J) in array `X`, and the number of columns equals the number of PCs (A);
- `MMULT` is a similar standard worksheet function (Section 7.2.6).

Example

An example is given in worksheet PLS1 and shown in Fig. 8.8.

`LoadingsPLS` is an array function, which must be completed by **CTRL+SHIFT+ENTER**.

8.4.4 `WLoadingsPLS`

Performs decomposition of matrices **X** and **Y** using the method of PLS (Eq. 8.2) and then returns an array that presents the loading weights values **W**.

Syntax

```
WLoadingsPLS (X, Y [, PC] [, CentWeightX] [, CentWeightY])
```

`X` is the array of **X**-values (calibration set);

`Y` is the array of **Y**-values (calibration set);

	I	J	K	L	M	N	O	P	Q
3		1	2	3	4	5	6	7	8
4	PC1	=TRANSPOSE(LoadingsPLS(Xcal,YcalA,5,1,1))							
5	PC2								
6	PC3								
7	PC4								
8	PC5								
9									

Figure 8.8 Example of `LoadingsPLS` function.

`PC` is an optional argument (integer), which defines the number of PLS components (A), used in the PLS decomposition;

`CentWeightX` is an optional argument (integer) that indicates whether centering and/or scaling for matrix **X** is done;

`CentWeightY` is an optional argument (integer) that indicates whether centering and/or scaling for matrix **Y** is done.

Remarks

- Array `Y` must have only one column ($K = 1$);
- Arrays `Y` and `X` must have the same number of rows (I);
- The result is an array (matrix) where the number of rows equals the number of columns (J) in array `X` and the number of columns equals the number of PCs (A);
- **`MMULT`** is a similar standard worksheet function (Section 7.2.6).

Example

An example is given in worksheet PLS1 and shown in Fig. 8.9.

`WLoadingsPLS` is an array function, which must be completed by **CTRL+SHIFT+ENTER**.

8.4.5 `QLoadingsPLS`

Performs decomposition of matrices **X** and **Y** using the method of PLS (Eq. 8.2) and then returns an array that presents the loadings values **Q**.

Syntax

`QLoadingsPLS` `(X, Y [, PC] [, CentWeightX] [, CentWeightY])`

`X` is the array of **X**-values (calibration set);

`Y` is the array of **Y**-values (calibration set);

	I	J	K	L	M	N	O	P	Q
12		1	2	3	4	5	6	7	8
13	PC1	=TRANSPOSE(WLoadingsPLS(Xcal,YcalA,5))							
14	PC2								
15	PC3								
16	PC4								
17	PC5								
18									

Figure 8.9 Example of **`WLoadingsPLS`** function.

`PC` is an optional argument (integer), which defines the number of PLS components (A), used in the PLS decomposition;

`CentWeightX` is an optional argument (integer) that indicates whether centering and/or scaling for matrix **X** is done;

`CentWeightY` is an optional argument (integer) that indicates whether centering and/or scaling for matrix **Y** is done.

Remarks

- Array `Y` must have only one column ($K = 1$);
- Arrays `Y` and `X` must have the same number of rows (I);
- The result is an array (vector) where the number of columns equals the number of PCs (A) and the number of rows equals the number of responses K, that is, 1;
- **`MMULT`** is a similar standard worksheet function (Section 7.2.6).

Example

An example is given in worksheet PLS1 and shown in Fig. 8.10.

`QLoadingsPLS` is an array function, which must be completed by **CTRL+SHIFT+ENTER**.

8.5 PLS2

8.5.1 `ScoresPLS2`

Performs decomposition of matrices **X** and **Y** using the method of PLS2 (Eq. 8.2) and then returns an array that presents the PLS2 score values $\mathbf{T}_{new}$ calculated for matrix $\mathbf{X}_{new}$.

Syntax

`ScoresPLS2``(X, Y [, PC] [, CentWeightX] [, CentWeightY] [, Xnew] )`

	I	J	K	L	M	N	O	P
20		A						
21	PC1	=TRANSPOSE(QLoadingsPLS(Xcal,YcalA,5,1,1))						
22	PC2							
23	PC3							
24	PC4							
25	PC5							
26								

Figure 8.10 Example of **`QLoadingsPLS`** function.

X is the array of **X**-values (calibration set);

Y is the array of **Y**-values (calibration set);

PC is an optional argument (integer), which defines the number of PLS components (A), used in the PLS2 decomposition;

CentWeightX is an optional argument (integer) that indicates whether centering and/or scaling for matrix **X** is done;

CentWeightY is an optional argument (integer) that indicates whether centering and/or scaling for matrix **Y** is done;

Xnew is an optional argument that presents an array of new values $\mathbf{X}_{new}$ (test set) for which PLS2 score values $\mathbf{T}_{new}$ are calculated;

Ynew is an optional argument that presents an array of new values $\mathbf{Y}_{new}$ (test set) for which PLS2 score values $\mathbf{T}_{new}$ are calculated.

Remarks

- Arrays **Y** and **X** must have the same number of rows (I);
- Arrays Xnew and **X** must have the same number of columns (J);
- If argument Xnew is omitted, it is assumed to be the same as **X**, and thus the calibration PLS2 score values **T** are returned;
- The result is an array (matrix) where the number of rows equals the number of rows in array Xnew and the number of columns equals the number of PCs (A);
- **TREND** is a similar standard worksheet function (Section 7.2.7).

Example

An example is given in worksheet PLS2 and shown in Fig. 8.11.

ScoresPLS2 is an array function, which must be completed by **CTRL+SHIFT+ENTER**.

8.5.2 UScoresPLS2

Performs decomposition of matrices **X** and **Y** using the method of PLS2 (Eq. 8.2) and then returns an array that presents the PLS score values $\mathbf{U}_{new}$ calculated for matrices $\mathbf{X}_{new}$ and $\mathbf{Y}_{new}$.

Syntax

```
UScoresPLS2 (X, Y, [, PC] [, CentWeightX] [, CentWeightY]
[, Xnew] [, Ynew])
```

X is the array of **X**-values (calibration set);

Y is the array of **Y**-values (calibration set);

	A	B	C	D	E	F	G	H
3			PC1	PC2	PC3	PC4	PC5	
4	calibration	1	3.384	0.029	0.065	-0.018	-0.025	
5		2	0.700	0.388	-0.082	-0.041	-0.017	
6		3	-0.679	0.183	0.127	0.058	0.024	
7		4	2.415	-0.210	0.029	0.001	0.046	
8		5	-0.419	0.081	-0.057	0.047	-0.023	
9		6	-2.529	0.161	0.043	-0.043	0.036	
10		7	-1.666	-0.236	0.099	-0.009	-0.058	
11		8	-0.286	-0.089	-0.161	0.040	0.001	
12		9	-0.920	-0.306	-0.063	-0.035	0.016	
13	test	10	=ScoresPLS2(Xcal,Ycal,5,1,1,Xtst)					
14		11						
15		12						
16		13						
17		14						
18								

Figure 8.11 Example of **ScoresPLS2** function.

PC is an optional argument (integer), which defines the number of PLS components (A), used in the PLS2 decomposition;

CentWeightX is an optional argument (integer) that indicates whether centering and/or scaling for matrix **X** is done;

CentWeightY is an optional argument (integer) that indicates whether centering and/or scaling for matrix **Y** is done;

Xnew is an optional argument that presents an array of new values $\mathbf{X}_{new}$ (test set) for which PLS2 score values $\mathbf{U}_{new}$ are calculated;

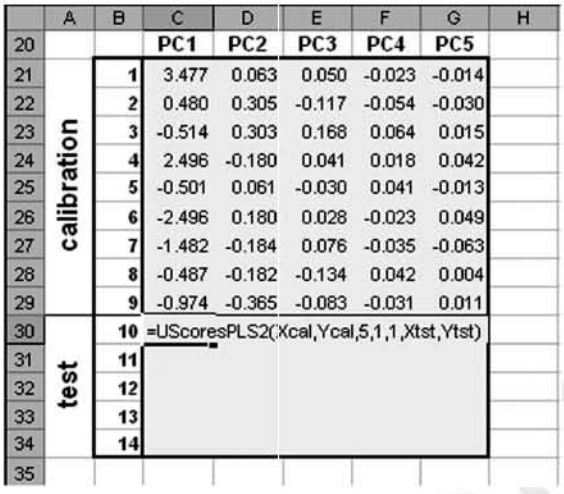
Ynew is an optional argument that presents an array of new values $\mathbf{Y}_{new}$ (test set) for which PLS2 score values $\mathbf{U}_{new}$ are calculated.

Example

An example is given in worksheet PLS2 and shown in Fig. 8.12.

Remarks

- Arrays **Y** and Ynew must have the same number of columns (K);
- Arrays **Y** and **X** must have the same number of rows (I);
- Arrays Xnew and **X** must have the same number of columns (J);
- If argument Xnew is omitted, it is assumed to be the same as **X**, and thus the calibration PLS2 score values **U** are returned;
- If argument Ynew is omitted, it is assumed to be the same as **Y**, and thus the PLS2 calibration score values **U** are returned;
- The result is an array (matrix) where the number of rows equals the number of rows in array Xnew and the number of columns equals the number of PCs (A);
- **TREND** is a similar standard worksheet function (Section 7.2.7).



			PC1	PC2	PC3	PC4	PC5
calibration	1		3.477	0.063	0.050	-0.023	-0.014
	2		0.480	0.305	-0.117	-0.054	-0.030
	3		-0.514	0.303	0.168	0.064	0.015
	4		2.496	-0.180	0.041	0.018	0.042
	5		-0.501	0.061	-0.030	0.041	-0.013
	6		-2.496	0.180	0.028	-0.023	0.049
	7		-1.482	-0.184	0.076	-0.035	-0.063
	8		-0.487	-0.182	-0.134	0.042	0.004
	9		-0.974	-0.365	-0.083	-0.031	0.011
test	10	=UScoresPLS2(Xcal,Ycal,5,1,1,Xtst,Ytst)					
	11						
	12						
	13						
	14						

Figure 8.12 Example of `UScoresPLS2` function.

`UScoresPLS2` is an array function, which must be completed by **CTRL+SHIFT+ENTER**.

8.5.3 `LoadingsPLS2`

Performs decomposition of matrices **X** and **Y** using the method of PLS2 (Eq. 8.2) and then returns an array that presents the loadings values **P**.

Syntax

`LoadingsPLS2 (X, Y [, PC] [, CentWeightX] [, CentWeightY])`

`X` is the array of **X**-values (calibration set);

`Y` is the array of **Y**-values (calibration set);

`PC` is an optional argument (integer), which defines the number of PLS components (A), used in the PLS2 decomposition;

`CentWeightX` is an optional argument (integer) that indicates whether centering and/or scaling for matrix **X** is done;

`CentWeightY` is an optional argument (integer) that indicates whether centering and/or scaling for matrix **Y** is done.

Remarks

- Arrays `Y` and `X` must have the same number of rows (I);

	I	J	K	L	M	N	O	P	Q
3		1	2	3	4	5	6	7	8
4	PC1	=TRANSPOSE(LoadingsPLS2(Xcal,Ycal,5,1,1))							
5	PC2								
6	PC3								
7	PC4								
8	PC5								
9									

Figure 8.13 Example of **LoadingsPLS2** function.

- The result is an array (matrix) where the number of rows equals the number of columns (J) in array **X** and the number of columns equals the number of PCs (A);
- **MMULT** is a similar standard worksheet function (Section 7.2.6).

Example

An example is given in worksheet PLS2 and shown in Fig. 8.13.

LoadingsPLS2 is an array function, which must be completed by **CTRL+SHIFT+ENTER**.

8.5.4 WLoadingsPLS2

Performs decomposition of matrices **X** and **Y** by using the method PLS2 (Eq. 8.2) and returns the loadings weights values **W**.

Syntax

```
WLoadingsPLS2 (X, Y [, PC] [, CentWeightX] [, CentWeightY])
```

X is the array of **X**-values (calibration set);

Y is the array of **Y**-values (calibration set);

PC is an optional argument (integer), which defines the number of PLS components (A), used in the PLS2 decomposition;

CentWeightX is an optional argument (integer) that indicates whether centering and/or scaling for matrix **X** is done;

CentWeightY is an optional argument (integer) that indicates whether centering and/or scaling for matrix **Y** is done.

Remarks

- Arrays **Y** and **X** must have the same number of rows (I);
- The result is an array (matrix) where the number of rows equals the number of columns (J) in array **X** and the number of columns equals the number of PCs (A);

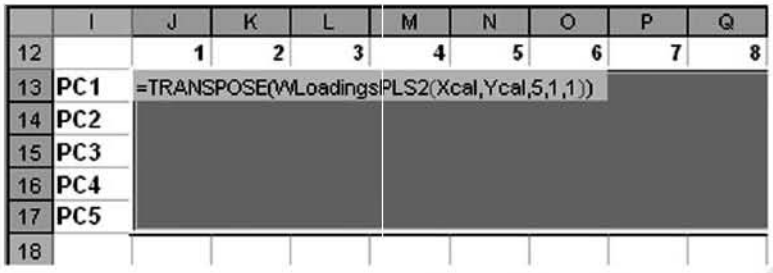


Figure 8.14 Example of `WLoadingsPLS2` function.

- **MMULT** is a similar standard worksheet function (Section 7.2.6).

Example

An example is given in worksheet PLS2 and shown in Fig. 8.14.

`WLoadingsPLS2` is an array function, which must be completed by **CTRL+SHIFT+ENTER**.

8.5.5 `QLoadingsPLS2`

Performs decomposition of matrices **X** and **Y** using the method of PLS2 (Eq. 8.2) and then returns the loadings values **Q**.

Syntax

```
QLoadingsPLS2 (X, Y [, PC] [, CentWeightX] [, CentWeightY])
```

`X` is the array of **X**-values (calibration set);

`Y` is the array of **Y**-values (calibration set);

`PC` is an optional argument (integer), which defines the number of PLS components (A), used in the PLS2 decomposition;

`CentWeightX` is an optional argument (integer) that indicates whether centering and/or scaling of matrix **X** is done;

`CentWeightY` is an optional argument (integer) that indicates whether centering and/or scaling of matrix **Y** is done.

Remarks

- Arrays `Y` and `X` must have the same number of rows (I);
- The result is an array (matrix) where the number of columns equals the number of PCs (A) and the number of rows equals the number of columns in array `Y`, that is, the number of responses K;
- **MMULT** is a similar standard worksheet function (Section 7.2.6).

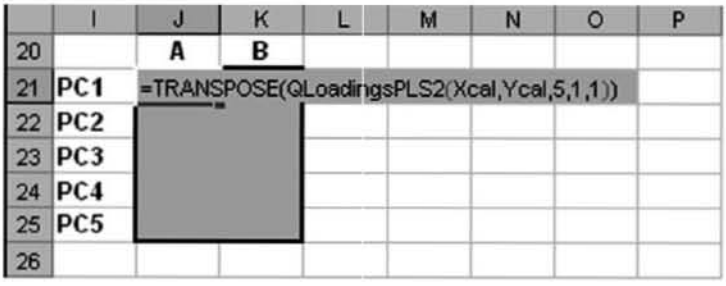


Figure 8.15 Example of function `QLoadingsPLS2`.

Example

An example is given in worksheet PLS2 and shown in Fig. 8.15.

`QLoadingsPLS2` is an array function, which must be completed by **CTRL+SHIFT+ENTER**.

8.6 ADDITIONAL FUNCTIONS

8.6.1 `MIdent`

Returns a square identity matrix

Syntax

`MIdent(Size)`

`Size` is an integer, which defines matrix dimension.

Remarks

- The result is a square array (matrix) with the number of rows and columns equal to `Size`;
- No similar standard worksheet function can be found.

Example

An example is given in worksheet Plus and shown in Fig. 8.16.

`MIdent` is an array function, which must be completed by **CTRL+SHIFT+ENTER**.

8.6.2 `MIdentD2`

Returns a two-diagonal rectangular matrix of zeros and ones, which is a result of cutting a row-block from an identity matrix.

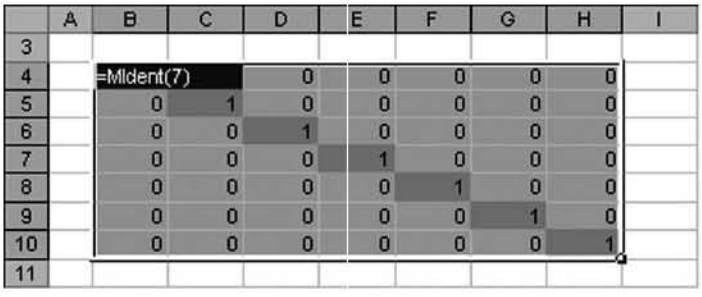


Figure 8.16 Example of **MIdent** function.

Syntax

MIdentD2(Size, CutFrom, CutOff)

Size is an integer (**Size** > 1), which defines the dimension of the initial identity matrix;

CutFrom is an integer (**CutFrom** > 0), which defines the row number where the cut starts;

CutOff is an integer (**CutOff** ≥ 0), which defines the number of rows to be deleted from the initial identity matrix.

Remarks

- The result is a rectangular array (matrix) where the number of rows equals **Size-CutOff** and the number of columns equals **Size**;
- When (**CutFrom** + **CutOff** − 1) > **Size**, the function returns #VALUE!;
- No similar standard worksheet function can be found.

Example

An example is given in worksheet Plus and shown in Fig. 8.17.

MIdentD2 is an array function, which must be completed by **CTRL+SHIFT+ENTER**.

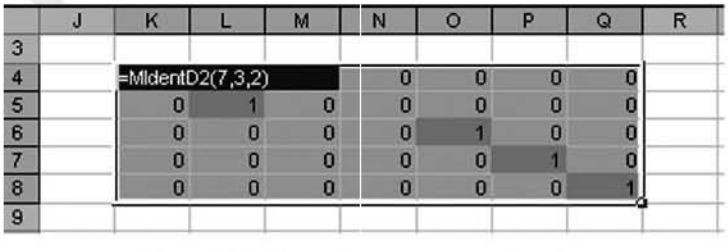


Figure 8.17 Example of **MIdentD2** function.

8.6.3 MCutRows

Returns a matrix with removed rows.

Syntax

MCutRows(X, CutFrom, CutOff)

X is an array (matrix) to be cut;

CutFrom is an integer (**CutFrom** > 0), which defines the row number where cut starts;

CutOff is integer (**CutOff** ≥ 0), which defines the number of rows that should be deleted from array **X**.

Remarks

- The function calculates the initial dimension of array **X**: nRows and nColumns. The result is a rectangular array (matrix) where the number of rows equals nRows−**CutOff** and the number of columns equals nColumns;
- When (**CutFrom** + **CutOff** − 1) > nRows, the function returns #VALUE!;
- **OFFSET** is a similar standard worksheet function (Section 7.2.4).

Example

An example is given in worksheet Plus and shown in Fig. 8.18.

MCutRows is an array function, which must be completed by **CTRL+SHIFT+ENTER**.

8.6.4 MTrace

Returns the trace of a matrix (array).

Syntax

MTrace(X)

	A	B	C	D	E	F	G	H	I	J	K
21		1	2	3	4	5	6	7	8	9	10
22	1	{=MCutRows(Xcal,7,1)			0.63	0.67	0.67	0.71	0.76	0.82	0.84
23	2	0.31	0.34	0.34	0.37	0.43	0.46	0.48	0.47	0.53	0.53
24	3	0.28	0.27	0.31	0.29	0.32	0.35	0.36	0.36	0.40	0.41
25	4	0.42	0.46	0.48	0.54	0.55	0.57	0.63	0.66	0.67	0.71
26	5	0.28	0.29	0.30	0.29	0.29	0.34	0.32	0.35	0.37	0.39
27	6	0.13	0.13	0.13	0.15	0.15	0.18	0.16	0.12	0.15	0.17
28	8	0.29	0.26	0.30	0.29	0.30	0.32	0.32	0.32	0.35	0.35
29	9	0.23	0.25	0.23	0.24	0.25	0.29	0.24	0.28	0.27	0.28
30											

Figure 8.18 Example of **MCutRows** function.

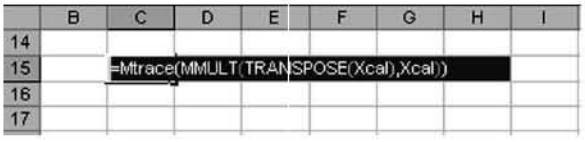


Figure 8.19 Example of **MTrace** function.

X is a square array (matrix).

Remarks

- **X** must be a square matrix;
- **MDETERM** is a similar standard worksheet function (Section 7.2.5).

Example

An example is given in worksheet Plus and shown in Fig. 8.19.
MTrace is not as an array formula and can be completed by **ENTER**.

CONCLUSION

Chemometrics Add-In provides many possibilities to calculate various scores and loadings for projection methods using original data. Furthermore, employing these results, a typical graphic representation used in multivariate data analysis can be obtained, among other score plots and loading plots. Such an approach also provides possibilities for the construction of calibration models, calculation of root mean square error of calibration (RMSEC), and root mean square error of prediction (RMSEP), etc.