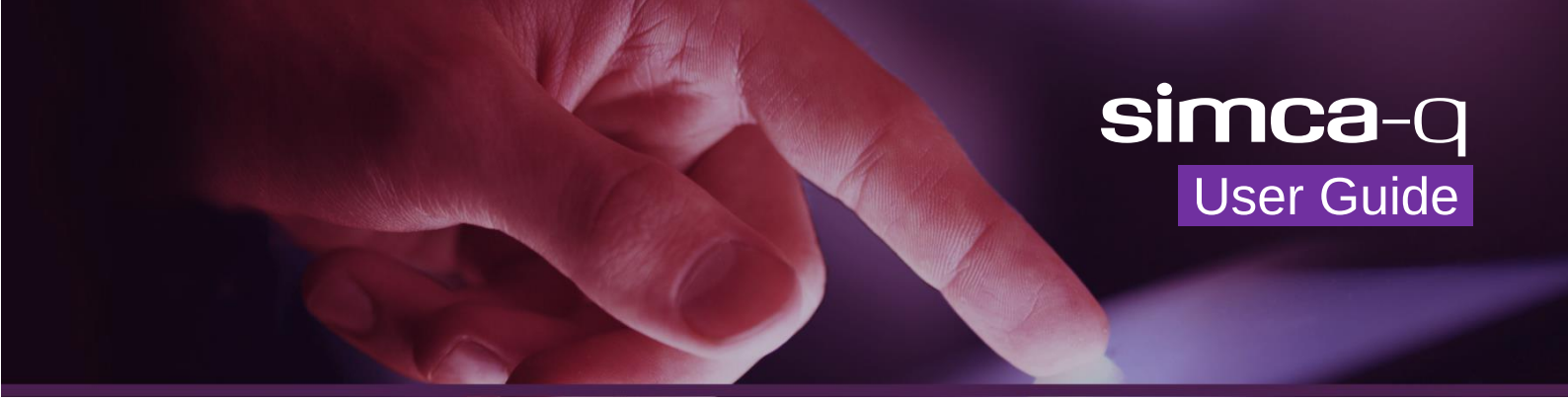


SIMCA-Q User Guide

July 8, 2016

The main purpose of this document is to describe the functionality and interfaces of SIMCA®-Q embedded solution.

SIMCA-Q is not a “plug and play” product, but is meant to be encapsulated within another product. The SIMCA-Q component is a DLL file with a C interface and a COM interface.

Basic knowledge of MVA is necessary in order to understand the terminology used in SIMCA-Q. We therefore recommend attending a course in MVA before starting the development.

Purpose and functionality

SIMCA-Q is intended to serve as a calculation engine for data and to produce MVA predictions on request. It is event-driven, i.e. it only makes predictions upon request, and is otherwise idle. Predictions can be requested for a project usp file created with SIMCA or other products from the Umetrics™ Suite of Data Analytics Solutions. All communication with SIMCA-Q is handled through a given interface.

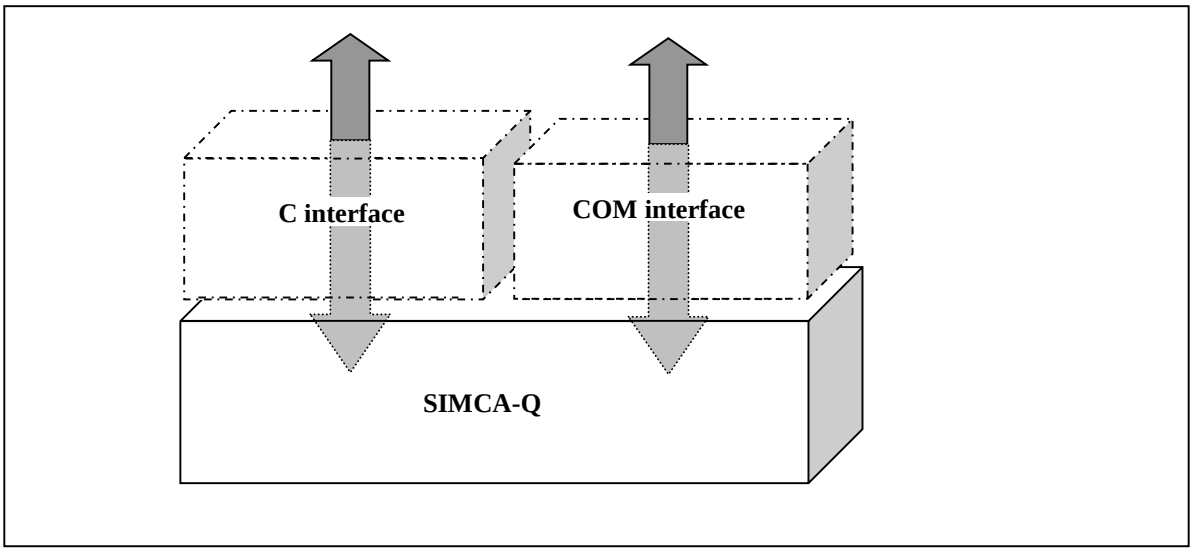
SIMCA-Q can perform predictions on continuous projects and, depending on licensing, perform predictions on batch projects and create continuous models.

Interfaces

SIMCA-Q has two main interfaces: a C interface and a COM interface.

The interface for SIMCA-Q consists of several functions. An overview of the functions can be found in ‘SIMCA-Q Interface Description.pdf’. Detailed descriptions are found in the generated help from the header files.

Note: Necessary header files and import libraries are delivered with the SIMCA-Q DLL.



Compatibility

SIMCA-Q 14.1 is compatible with projects created by SIMCA 13 - 14.1 or other products from the Umetrics Suite of Data Analytics Solutions of version 13 to 14.1. Projects created with earlier versions must first be converted to a version 13 to 14.1 project through SIMCA 13 to 14.1.

License file

SIMCA-Q cannot be used unless accompanied by a license file. The license file controls how SIMCA-Q may be used, e.g. the number of projects that can be loaded, etc.

The license file will default be installed in the ProgramData directory (typical path: C:\ProgramData\Umetrics\SIMCA-Q), but the file can be moved and the path can be overridden by a function call to SIMCA-Q.

Resource ID

Resource IDs larger than 9999 are reserved by SIMCA-Q.

Error handling

All functions return an error code upon failure. An error message can be retrieved from the error code and all errors are logged in the SIMCA-Q internal log file. The log will contain information regarding where and why the error occurred.

By default, the log file is named **SIMCA-Q.log** and will be located in the Application Data directory. A typical path is C:\Documents and Settings\UserName\Application Data\Roaming\Umetrics\SIMCA-Q\14.1. Both the name and location of the log file can be changed through a function call.

There is also a function available to disable the creation of the log file, as well as a function to supply SIMCA-Q with a function pointer (a callback method) to which the log information will be sent. The latter function is only available in the C interface.

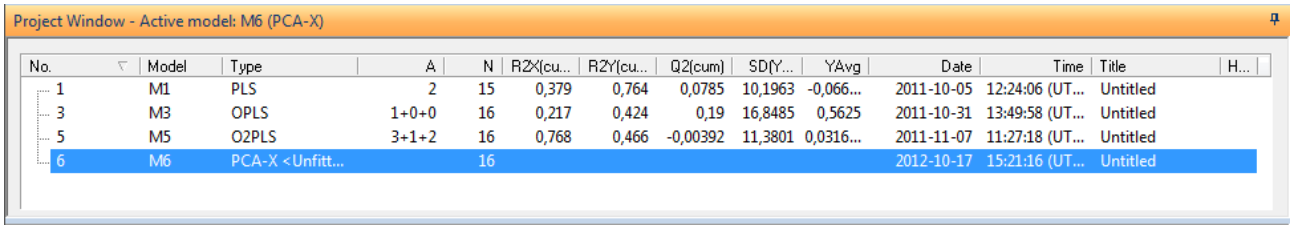
Unicode

SIMCA-Q supports Unicode characters. UTF-8 is used as character encoding which means that all strings that are sent to SIMCA-Q have to be in UTF-8 encoding. If only the standard English alphabet characters are used the encoding is not relevant.

Models and model number

A project can contain one or several models of the types PCA, PLS, OPLS and O2PLS including descriptive analysis and class/phase models. In this document O2PLS is included when referring to OPLS.

The model number is required as input for some of the functions found in the interface. Note that the model number is not identical to the model index. For example, if one or more models have been deleted from the project, the model index and the model number will differ. The model number is the number of the model in SIMCA, displayed in the “No.” column of the project window.



No.	Model	Type	A	N	R2X(cu...	R2Y(cu...	Q2(cum)	SD(Y...	YAvg	Date	Time	Title	H...
1	M1	PLS	2	15	0,379	0,764	0,0785	10,1963	-0,066...	2011-10-05	12:24:06 (UT...	Untitled	
3	M3	OPLS	1+0+0	16	0,217	0,424	0,19	16,8485	0,5625	2011-10-31	13:49:58 (UT...	Untitled	
5	M5	O2PLS	3+1+2	16	0,768	0,466	-0,00392	11,3801	0,0316...	2011-11-07	11:27:18 (UT...	Untitled	
6	M6	PCA-X < Unfitt...		16						2012-10-17	15:21:16 (UT...	Untitled	

Figure 1: In the image above we have four models with the model numbers 1, 3, 5 and 6 since model numbers 2 and 4 have been deleted. Model number 6 has not yet been fitted.

Batch project structure

A batch project consists of one or several Batch Models (BM). Each BM has one or several Batch Evolution Models (BEM), one for each modeled phase, and may also contain one or several Batch Level Models (BLM). This is illustrated as follows in SIMCA.

Project Window - Active model: M1:chip (PLS-Class)

No.	Model	Type	A	N	R2X(cum)	R2Y(cum)	Q2(cum)	SD(YR...	YAvg	Date	Time	Title	Hi...
BM1										2012-05-16	11:13:08 (UTC+1)	Untitled	
BEM													
1	M1:chip	PLS-Class(chip)	8	354	0,832	0,863	0,849	0,142896	0,602166	2012-05-16	11:13:03 (UTC+1)	Untitled	
2	M2:acid	PLS-Class(acid)	8	345	0,749	0,533	0,387	0,310752	0,653526	2012-05-16	11:13:05 (UTC+1)	Untitled	
3	M3:cook	PLS-Class(cook)	7	1657	0,799	0,943	0,927	0,38364	2,75045	2012-05-16	11:13:06 (UTC+1)	Untitled	
4	M4:blbk	PLS-Class(blbk)	7	269	0,726	0,883	0,844	0,0960068	0,443989	2012-05-16	11:13:07 (UTC+1)	Untitled	
5	M5:blow	PLS-Class(blow)	8	106	0,877	0,954	0,846	0,0220466	0,16069	2012-05-16	11:13:08 (UTC+1)	Untitled	
BLM													
6	M6	PCA-X	2	10	0,382		-0,186			2012-05-16	11:13:37 (UTC+1)	Untitled	
14	M14	PLS	6	10	0,768	0,987	0,623	0,774301	30,215	2012-12-06	09:07:45 (UTC+1)	Untitled	
BM2										2012-05-16	11:15:43 (UTC+1)	Untitled	
BEM													
9	M9:chip	PLS-Class(chip)	8	354	0,832	0,863	0,849	0,142896	0,602166	2012-05-16	11:15:43 (UTC+1)	Untitled	
10	M10:acid	PLS-Class(acid)	8	345	0,749	0,533	0,387	0,310752	0,653526	2012-05-16	11:15:43 (UTC+1)	Untitled	
11	M11:blbk	PLS-Class(blbk)	7	269	0,726	0,883	0,844	0,0960068	0,443989	2012-05-16	11:15:43 (UTC+1)	Untitled	
12	M12:blow	PLS-Class(blow)	8	106	0,877	0,954	0,846	0,0220466	0,16069	2012-05-16	11:15:43 (UTC+1)	Untitled	
BLM													
13	M13	PCA-X	2	10	0,378		-0,118			2012-05-16	11:17:15 (UTC+1)	Untitled	

SIMCA-Q uses the same structure as SIMCA. From the project object, the BatchModel object is retrieved, from the BatchModel object, both the BEM objects and the BLM objects can be retrieved.

Call sequence for predictions on continuous projects

The call sequence for SIMCA-Q is as follows:

1. Open a project, with the full path to the usp file as input
2. Load a model, with the model number as input
3. Get info from the model, e.g. T, DModX, etc.
4. Set data for prediction
5. Perform the prediction
6. Get results from prediction, e.g. TPS, DModXPS, etc.
7. Close the project

Steps 4-6, as well as steps 2-6, may be repeated as desired.

Call sequence for predictions on batch projects

The call sequence for a batch project with batch level models is as follows:

1. Open a project, with the full path to the usp file as input
2. Load a BatchModel
3. Load the batch level model to get predictions from
4. For each batch evolution model, set data for prediction
5. Set batch condition data for the batch level model.
6. Perform the prediction
7. Get results from prediction, e.g. TPS, DModXPS etc.
8. Close project

4-7 can be repeated as well as 2-7.

For batch projects without batch level model, the call sequence is the same as for continuous projects.

Performing a prediction

Predictions can be performed on one observation at a time, or on several observations at once. If performance is of importance, the prediction speed per observation will be much faster if several observations are bundled together.

Order of input data

The order of the data must be the same as that returned by the function **GetVariablesForPrediction**.

Quantitative vs. qualitative functions

There are two different functions to specify data with, one for quantitative data and one for qualitative data. Both functions can be used for both kinds of data, as illustrated in the following example.

Model M1 contains the following variables: Var1, Var2, Var3 (a Qualitative variable with settings A1, B2, and C3) and Var4.

	Primar	2	3	Qualita	5
Primar		Var1	Var2	Var3	Var4
2	1	1,1	8,8	B2	5,3
3	2	1,4	9,5	A1	5,5
4	3	1,2	9,3	C3	5,7

Figure 2: This is the data to use as predictionset

GetVariablesForPrediction will return Var1, Var2, Var3, and Var4.

There are three different ways to specify the data:

1. Set quantitative data as quantitative data and qualitative as qualitative.
   This is the most straightforward approach. Note that when setting data, the index of the variable should always be the index returned by **GetVariablesForPrediction**. Quantitative data will be set for variable index 1, 2 and 4 and qualitative data for variable index 3.
2. Set all as quantitative data.
   Even if there are qualitative data, one can specify this as quantitative. For the first observation we have the setting B2 for the qualitative variable. B2 has the index 2 among the qualitative settings for Var3 (all settings in order can be retrieved via the function **GetQualitativeSettings** for the Var3 variable). To set B2 as a quantitative value, use the index, in this case 2.
3. Set all as qualitative.
   The quantitative data can be set as qualitative simply by making it into a string, e.g. the value 1.1 is represented by the string “1.1”. Note that this approach will significantly slow down performance.

Lagged variables

If the model contains lagged variables, you have the option to set data for these. If only one or a few observations are to be predicted at a time, it may be necessary to send data for the lagged variables. However, if many observations are to be predicted at once, leaving the lags as missing is not a problem. When lags are not supplied, they will be set as missing for the first time points.

If the data are to be specified, this is done with the functions **SetQuantitativeLagData** and **SetQualitativeLagData**. Specify data for each variable up to the largest lag step.

	Primar	2	3
Primar	Primary ID	Var1	Var2
2	-2	1	
3	-1	1,3	9,2
4	0	0,9	9,1

Figure 3: Assume that the model from Figure 2 above contains three variables that have been lagged: Var1.L1, Var1.L3 and Var2.L2. In order to set data for these variables there must exist three historical values for Var1 and two historical values for Var2, as illustrated in this table.

Filtered data

If the model contains filtered data, SIMCA-Q will automatically apply the same filter to the predictions. **GetVariablesForPrediction** will return the original variables that the filter was created from. The user should treat these projects like any other regular project.

Missing values

If a variable lacks data, or has invalid data, it should be given the value that represents a missing value. This must be done for each observation that partially lacks data. However, an observation that lacks data entirely should not be set at all.

The value representing a missing value in SIMCA-Q is accessible through the function **GetMissingValue**.

Note that if the data to be predicted is dominated by missing values, the prediction will be very time consuming.

Performing a prediction for a batch model

Predictions will be performed on one batch at a time.

Predictions for a batch level model

To perform a prediction for a batch level model the batch level model is used to get a **PrepareBatchPrediction** object. The **PrepareBatchPrediction** object is then used to set data for all batch evolution models, one at a time. If batch conditions are included in the batch level model they will also be set in the **PrepareBatchPrediction** object, one value for each batch condition variable. When all data is set, **GetBatchPrediction** is called and a **BatchPrediction** object is returned. From that the predictions from the batch evolution models and the batch level models can be retrieved.

Predictions on ONLY the batch evolution model(s)

To perform predictions on only the batch evolution model, the same approach as continuous projects should be used, i.e. open the model and get the **PreparePrediction** object from that model.

Lack of data for phases

Data might not have been set for all phases before the call to **GetBatchPredictions**(). If one of the phases that lack data is part of the batch level model, SIMCA-Q will use the "average phase" values for the batch level predictions. In that way predictions from the batch level can be requested from SIMCA-Q. However, predictions for the batch evolution model can only be requested for the phases that were provided to **PrepareBatchPrediction**.

Conditional delete

When a project is created in SIMCA it can contain conditional delete statements. Due to these the effect could be that an observation level model lacks data and is not predicted. In that case the batch level predictions will use the "average phase" values for the batch level predictions. No predictions for the batch evolution model can be requested for that model.

Error when predicting for the batch evolution model

If an error occurs when predicting a batch evolution model, e.g. erroneous input, the phase connected with the model will be treated as a phase that lacks data. The reason for using average batch will be found in the log file.

Local centering

For a project, variables might have been centered by importing a local centering file when creating the project in SIMCA. SIMCA-Q will perform this centering automatically. The function **GetLocalCenteringInfo** will return the variables that are centered and their default values for a given model that is connected with a phase. If other values should be used use the function **SpecifyLocalCentering** to set those values. Only variables that are UVN scaled will be centered.

If a variable in a batch project was centered while creating the project in SIMCA and excluded in the batch evolution models of the project but part of the batch level, that variable will not be reported by SIMCA-Q since it is limited to only report the centering values connected with the batch evolution models.

Retrieving results

Most results can be requested both from the model and from the prediction. For example, to retrieve scores from the model, use the **GetT** function from the Model object. To get scores from the prediction, use the **GetTPS** function from the Prediction object. All vectors from the prediction are identified by a 'PS' in the name.

The results are retrieved in the same way as for a continuous project, but for the batch evolution models the aligned results can also be retrieved. The same goes for the prediction, the aligned PS results are retrieved from the batch evolution prediction object.

VectorData

All vectors returned from SIMCA-Q will be in the form of a **VectorData** object. The **VectorData** object consists of the data matrix plus the column and row names. For example, the following score vector will have a vector data object that contains a matrix (2 columns, 5 rows), column names (M4.t[1], M4.t[2]) and row names (1, 2, 3, 4, and 5).

General List [M4]

	1	2	3
1	Obs ID (Primary)	M4.t[1]	M4.t[2]
2	1	-1,31914	0,0857188
3	2	2,38607	-1,62632
4	3	-0,462383	-2,88174
5	4	-2,90623	-0,501176
6	5	0,508198	-1,29503

If the data comes from a prediction, e.g. TPS, the observations will be named in the following pattern: From Q_1_Obs1, From Q_2_Obs2, and so on.

Component number

When requesting data for a vector, you may sometimes need to send the component number as input. Some functions are valid even if there are no components in the model, while other functions will fail for component number zero (0). For an OPLS model, some vectors are only valid for the last predictive component. Whether or not a function is valid for a zero component model is documented in the header files and in the detailed function documentation.

Column

Qualitative variables will be expanded in the **VectorData** matrix to comprise one column for each setting. For example, when requesting XVar from the model described in Figure 1 on page 5, the **VectorData** will appear as follows:

General List [M1]

	1	2	3	4	5	6	7
1	Obs ID (Primary)	M1.XVar(Var1)	M1.XVar(Var2)	M1.XVar(Var3(A1))	M1.XVar(Var3(B2))	M1.XVar(Var3(C3))	M1.XVar(Var4)
2	1	1,1	8,8	9,93411e-009	1	9,93411e-009	5,3
3	2	1,4	9,5	1	9,93411e-009	9,93411e-009	5,5
4	3	1,2	9,3	9,93411e-009	9,93411e-009	1	5,7

Look at columns 4, 5 and 6 above. Notice how the qualitative variable, Var3, has been expanded into three columns for A1, B2 and C3 respectively. These columns will contain a 0 for settings not set, and a 1 for the active setting. To get the names, etc. of the expanded columns, use the functions beginning with **GetColumn** from the model: **GetColumnXSize**, **GetColumnXNameByIndex**, etc.

Contributions

Contributions can be requested from both the models and the predicted data.

When you request a contribution, SIMCA-Q requires the weights, component number and type of contribution (Score, DModX, and DModY) as input.

To be able to interpret the result returned from a request for contribution, e.g. to identify which variable each returned column is connected with, the column functions should be used.

Score contributions

There are two types of contributions that can be requested for scores: Scores Single Weight or Scores Multi Weight. For batch projects combined contributions can also be constructed.

Scores Single Weight
This contribution cannot be requested for a zero component model.

- **Weight**: If the model is a PCA model the weight must be Normalized, Raw, RX or P. For a PLS model the weight must be NoWeight, RX, P, WStar, VIP, CoeffCS or CoeffCSRaw. OPLS model can have the same weights as PLS and additionally the weight Po. If the weight is either CoeffCS or CoeffCSRaw, the y-variable must be given as input.
- **Component**: This contribution can only be calculated for a single component.
- **Observations**: To obtain the contribution showing the difference between two observations, the index of the reference observation and the index of the second observation must be provided. The contribution relative to the average of the X variables as reference can be retrieved by setting the index of the reference observation to zero. If only one observation exists in the prediction, the index values will be ignored. In this case, the average of the X variables is used as reference.

Scores Multi Weight
This contribution cannot be requested for a zero component model.

- **Weight**: If the model is a PCA model the weight must be P. For a PLS model the weight must be P or WStar. OPLS models have the same weight as PLS and additionally the weight Po. For OPLS models both weight P and Po can be used at the same time for different components.
- **Component list**: This contribution can be calculated for one or more components. A list containing the selected components is given as input.

- **Observations**: To obtain the contribution showing the difference between two observations, the index of the reference observation and the index of the second observation must be provided. The contribution relative to the average of the X variables as reference can be retrieved by setting the index of the reference observation to zero. If only one observation exists in the prediction, the index values will be ignored. In this case, the average of the X variables is used as reference.

Contribution for DModX

This contribution can be requested for a zero component model.

- **Weight**: If the model is a PCA model the weight must be Normalized or RX. For a PLS/OPLS model the weight must be Normalized, RX, VIP or CoeffCS. If the weight is CoeffCS, the y-variable must be given as input. Note! For a zero component model the only valid weight is Normalized, regardless of model type.
- **Component**: This contribution is the observation residuals, multiplied by the selected weight, after the specified number of components.
- **Observations**: Specifies the indexes of the observations in the model or predictions.

Contribution for DModY

This contribution can only be requested for a PLS or OPLS model with at least one component.

- **Weight**: The weight must be Normalized or RY.
- **Component**: This contribution is the observation residuals, multiplied by the selected weight, after the specified number of components.
- **Observations**: Specifies the indexes of the observations in the model or predictions.

Group contributions

Contributions can be requested for a group of observations. The only difference in the interface is that a list of observations is requested as input instead of a single observation.

Contribution scores SIMCA-Q vs. SIMCA

The following table describes the relationship between contributions in SIMCA-Q and SIMCA.

Contribution type	Weight	Contribution in SIMCA
ScoresSingleWeight	Normalized	Contribution Scores with weight Normalized.
ScoresSingleWeight	Raw	Contribution Scores with weight Raw.
ScoresSingleWeight	P	Contribution Scores with weight p
ScoresSingleWeight	Po (OPLS only)	Contribution Scores with weight po
ScoresSingleWeight	WStar (PLS and OPLS)	Contribution Scores with weight w*
ScoresSingleWeight	RX (PCA)	Contribution Scores with weight RX
ScoresSingleWeight	RX (PLS and OPLS)	No equivalent in SIMCA but not illegal
ScoresSingleWeight	CoeffCS (PLS and OPLS)	Contribution Scores with weight CoeffCS
ScoresSingleWeight	CoeffCSRaw (PLS and OPLS)	Contribution Y-predicted
ScoresSingleWeight	VIP	Contribution Scores with weight VIP
ScoresMultiWeight	P with 2 components	Contribution Scores with Weight pp
ScoresMultiWeight	P, Po with 2 components (OPLS only)	Contribution Scores with Weight ppo
ScoresMultiWeight	P with more than 2 consecutive components	Contribution Scores with weight PRange
ScoresMultiWeight	P with more than 2 non-consecutive components	No equivalent in SIMCA but not illegal
ScoresMultiWeight	WStar with 2 components (PLS and OPLS)	Contribution Scores with weight w*w*
ScoresMultiWeight	WStar with more than 2 consecutive components (PLS and OPLS)	Contribution Scores with weight W*Range

ScoresMultiWeight	WStar with more than 2 non-consecutive components (PLS and OPLS)	No equivalent in SIMCA but not illegal

Control Charts

Control charts are used to observe the data measured on a process or product over time and thereby detect process upsets, shifts, trends, etc.

With all control charts, the target and standard deviation can either be estimated or entered by the user.

Unlike SIMCA, there is no need to select the data, item, number and component. The values to be used will be put into a vector.

Shewhart Control Chart

Two types of Shewhart charts are available in SIMCA-Q: Mean/Range and Mean/StdDev.

The sample size must be smaller than the number of observations, and between 2 and 25.

Cusum Control Chart

This chart is used to detect a deviation from the target. The sample size must be smaller than the number of observations, and between 2 and 25.

GetHighCuSum() Retrieves the cumulative sum on the high-side difference and is used to detect a deviation from the target on the high side.

GetLowCuSum() Retrieves the cumulative sum on the low-side difference and is used to detect a deviation from the target on the low side.

EWMA Control Chart

EWMA stands for Exponentially Weighted Moving Averages.

The sample size must be smaller than the number of observations, and between 2 and 25.

The lambda can either be estimated or entered by the user. When not entered by the user, it will be estimated to a value that will minimize the error sum of squares.

Reduced SIMCA project (rusp)

A SIMCA project can be reduced in SIMCA. When the project is reduced one can choose what to remove from the project. The things that can be removed are; the model residuals and the datasets. When the datasets or the model residuals are removed, the project will get the extension rusp and can't be opened by SIMCA.

For a reduced SIMCA project not all functions are available.

Functions that fail when the datasets have been removed:

- All GetDataSet- functions
- GetXObs
- GetXObsRes
- GetXObsPred
- GetYObs
- GetYObsRes

For the evolution level of a batch project the following functions will also fail when the datasets have been removed:

- All GetAligned- functions
- All GetAlignedPS- functions
- GetT2Range

Functions that fail when the model residuals have been removed:

- GetDModY
- GetMPowX
- GetORisk
- GetORiskPooled
- GetPModX
- GetPModY
- GetYPredCV
- GetYPredCVErr
- GetYPredCVErrSE

- GetR2VXAdj
- GetR2VXAdjCum
- GetR2VYAdj
- GetR2VYAdjCum
- GetSerrL
- GetSerrU
- GetS2VX
- GetS2VY
- GetXVar
- GetXVarRes
- GetSerrLPS
- GetSerrUPS

For the evolution level of a batch project the following functions will also fail when the model residuals

are removed:

- GetContributionsSSW
- GetContributionsScoresSingleWeightGroup
- GetContributionsScoresMultiWeight
- GetContributionsScoresMultiWeightGroup
- GetContributionsScorePSSingleWeight
- GetContributionsScorePSSingleWeightGroup
- GetContributionsScorePSMultiWeight
- GetContributionsScorePSMultiWeightGroup

Initiating a project for modeling

There are two approaches to initiate a project for modeling in SIMCA-Q. The first is to create a new project and the other is to open an existing project.

If SIMCA-Q is shut down the project/s must be reinitialized.

Temporary directory

A project can be loaded or created with or without a *temporary directory*.

When a temporary directory is defined, all changes to the project will be saved to a temporary directory and not to the usp file itself. First when the user chooses to save, the changes will be saved to the usp file. This approach avoids the usp file from becoming corrupt. If something goes wrong when working with a project; that project can be reloaded with the recovery flag set or not. When the recovery flag is set SIMCA-Q will try to recover the incomplete project from the temporary directory. If the flag is not set SIMCA-Q will open the usp file in the state it had when it was last saved.

If a project is loaded or created without a temporary directory, every change in the project will be saved immediately to the usp file.

Create a new project

To create a new project, the Import interface will be used.

Import

The import procedure can be divided into the following steps.

- Initiate Import; call **InitImport()** with the path to the new project. The “Import Handle” received should be used to access all other Import function.
- Specify the dataset; minimum requirement is to call either **AddQuantitativeVariables()** or **AddQualitativeVariables()**. In this step the user specifies the data, observation IDs, variable IDs and missing value representation. If many variables are going to be added, it is a good idea to call **Reserve()** since this will minimize memory reallocation.
- Create project; call **FinishImport()** to create the project from the specified data. The project handle returned should be saved for later use with other SIMCA-QM functions. To close the project and release memory call **CloseProject()** with the project handle.

Dataset

Models can be created on data from one or several datasets. To import more than one dataset use the function **InitImportDataset()**.

Generated variables

From variables in a dataset one can generate new variables. SIMCA has several predefined functions for generated variables and the user can add their own functions as well. SIMCA-Q uses the same nomenclature as SIMCA for the generated variables.

An example:

If the dataset looks like this:

Primary ID	Obs. Sec. ID:1	Gr_Coffe	Inst_Coffe	Tea	Sweetner
1	Germany	90	49	88	19
2	Italy	82	10	60	2
3	France	88	42	63	4
4	Holland	96	62	98	32
5	Belgium	94	38	48	11
6	Luxembou	97	61	86	28

The variable Gr_Coffe will have the name v1 when generating a variable from it. Inst_Coffe will have the name v2 etc.

Primary ID	Obs. Sec. ID:1	Gr_Coffe	Inst_Coffe	Tea	Sweetner
		v1	v2	v3	v4
1	Germany	90	49	88	19
2	Italy	82	10	60	2
3	France	88	42	63	4
4	Holland	96	62	98	32
5	Belgium	94	38	48	11
6	Luxembou	97	61	86	28

If we want to generate a new variable named “New” that is Gr_Coffe + Tea we will use the expression: “v1+v3” and get the following result:

Primary ID	Obs. Sec. ID:1	Gr_Coffe	Inst_Coffe	Tea	Sweetner	New
1	Germany	90	49	88	19	178
2	Italy	82	10	60	2	142
3	France	88	42	63	4	151
4	Holland	96	62	98	32	194
5	Belgium	94	38	48	11	142
6	Luxembou	97	61	86	28	183

Workset

Worksets are created by calling **GetNewWorkset()** which returns a workset handle. This handle is then used as an argument to the various Workset-functions to specify the workset. The workset will initially be empty (no variables and no observations). Variables/observations must be added one by one or through a call to **CreateDefaultWorkset()**. When the specification is done, a call to **CreateModel()** will create a new unfitted model. Creation of a model from a handle does not invalidate a handle. The same workset handle can be used to create several models.

All handles obtained from **GetNewWorkset()** must be released by **ReleaseWorkset()** whether or not a model has been created from the handle.

Lagged variables

Just like SIMCA, SIMCA-Q will remove observations from the workset when it contains lagged variables, the first max number of lag observations are removed from the workset. In other words, if the workset contains one variable lagged 3 steps and one lagged 2 steps, the first three observations will be removed from the workset.

Fit

The fit interface includes the functionality to auto fit, change model type, delete a model, and ability to remove and calculate components of a model.

Filter

SIMCA-Q has the functionality to apply a range of spectral filters to the data found in a project. A filter is used to transform data in the dataset. The filtered variables will be saved in a new dataset in the existing project. Each filter has the same options and settings as the corresponding filter in SIMCA. Applying a filter is a multistep process and generally the following order must be used

1. Create a spectral or time series filter.
2. Add what type of filter to use
   The following filter types exist in SIMCA-Q
   - MSC – Multiplicative Signal Correction.
   - OSC – Orthogonal Signal Correction.
   - Row Center
   - Savitzky-Golay
   - EWMA - Exponentially Weighted Moving Average
   - SNV – Standard Normal Variate transformation.
   - Derivatives – 1st and 2nd Derivative.
   - WDS – Wavelet Denoise Spectral.
   - WCS – Wavelet Compression Spectral.
   - WDTS – Wavelet Denoise/Decimation Time Series.
     This is a time series filter and it is not applicable on batch projects.
3. Set the data to apply filter on. Both variables and observations.
4. For some filters, transformation and/or scaling can be set on all or selected parts of the data set. For more limitations, like when no/one Y must be selected as data, see the SIMCA filter documentation.
   - The default scaling is Center for X and Unit variance for Y. Scaling can be set on OSC filters.
   - The default transformation is None. Transformation can be set on OSC and WCS.
5. Set filter specific options and general filter settings, like including original data and naming the new project.
6. Start the filter process.
   - Some filters, like OSC, are interactive, so data and measurements can be retrieved during the filter process, allowing options to be fine-tuned.
7. Finish the filter process. Here the filtered data are included in a new dataset.

When using a wavelet, or wavelet combination filter there exist a number of wavelet functions to choose between and each function has its own selectable wavelet orders.

Wavelet function	Wavelet orders
Beylkin	No selection available
Coiflet	2, 3, 4, 5
Daubechies	4, 6, 8, 10, 12, 20, 50
Symmlet	4, 6, 8, 10
Biorthogonal1	1, 3, 5
Biorthogonal2	2, 4, 6, 8
Biorthogonal3	1, 3, 5, 7, 9
Biorthogonal4	No selection available
Biorthogonal5	No selection available
Biorthogonal6	No selection available

The chained filter (the spectral filter) is special in the sense that the filter process is split up in two or more parts. First, the first chosen filter and its settings are applied like that ordinary filter. Then the settings for the next filter are applied to the result from the first filter and so on. Notice that the wavelet filter will generate as many new variables as coefficients selected in the wavelet settings. Also, these new variables will be appended last in the dataset.

See the SIMCA User Guide for a deeper explanation of the different filters and their settings.

Generated variables in predictions

Generated variables can be a part of the project using SIMCA-Q. However, there are some limitations concerning which generated variables can be handled by SIMCA-Q. The following generated variables will not be handled by SIMCA-Q:

- Lags created using generated variables
- Generated variables with phase settings
- Model results (e.g. score)

If the variables were created using a plug-in DLL, the path to the directory containing the plug-in DLL must be provided in the **SetPluginPath** function. The function must be called before the first call to **OpenProject**. If the function is not called, the plug-in DLL is assumed to be in the same directory as the SIMCA-Q DLL.

Threading

SIMCA-Q can be used in different threads per project. This means that one thread can call **OpenProject** with one usp file and another thread can call **OpenProject** with a different usp file. The two threads will then work in parallel on the different projects. Performing operations on the same project from different threads is not guaranteed to be thread safe.

SIMCA-Q uses different threads internally to perform certain calculations. This can be turned on or off by a call to **UseMultiThreading**. Multi-threading is turned off by default and should remain off for small projects where the overhead needed to create several threads is too large.

Samples

A number of samples illustrating how to use SIMCA-Q from different environments are available from the website. Some of these, such as VB, C# and C++, encapsulate the C/COM interface in a higher level interface called EzQ, demonstrating how a higher level interface can be created.