Automating and extending comprehensive two-dimensional gas chromatography data processing by interfacing open-source and commercial software

Michael J Wilde*ab, Bo Zhaoac, Rebecca L. Cordell, Wadah Ibrahimbcd, Amisha Singapurbcd, Neil J. Greeningbcd, Chris E. Brightlingbcd, Salman Siddiquibcd, Paul S. Monksa, Robert C. Freeac

a School of Chemistry, b Department of Respiratory Sciences, University of Leicester, University Road, Leicester, LE1 7RH, UK. c Leicester NIHR Biomedical Research Centre, Glenfield Hospital, Groby Road, Leicester, LE3 9QP, UK.

Alongside supporting tables and figures cross-referenced in the main text, instructions for using the example folders and for creating command and batch files are given in the Supporting Information. The example folders contain files equivalent to those used in the exemplar methods and have been made available via a repository at https://github.com/rcfgroup/gc-automation.

SUPPORTING INFORMATION TABLE OF CONTENTS

TABLES
Table S1. S-2

FIGURES
Figure S1. S-5

EXAMPLE FOLDER INSTRUCTIONS
Before you begin S-6
Explanation of Example Folders S-9
Instructions S-10
1 Match Template folder
2 Export Match File folder
3 Apply Match File folder
Results S-11
1 Match Template folder
2 Export Match File folder
3 Apply Match File folder

FAMILIARISATION AND CREATION OF NEW INTERFACE FILES
Locating the command line interface S-14
Creating a command file S-14
Creating a batch file S-16
Integrating external scripts S-17
Accessing the command line in other software S-18

REFERENCES
Reference list S-19
## TABLES

Table S1. Commercial, bespoke and FOS chemometric software available for GC×GC data processing (extensive but not limited to)

| Type            | Main Function                          | Software / Package | Developed in | Description | Ref. |
|-----------------|----------------------------------------|--------------------|--------------|-------------|------|
| Commercial      | Multi-functional / workflow             | GC Image™ suite    | Java (GC Image) | All have advanced full-featured GUIs. All capable of common pre-processing steps based on proven methods including baseline correction, deconvolution, alignment (peak merging/matching) as well as visualisation, and comparative and statistical analysis. | -    |
|                 |                                        | ChromaTOF®        | C++ and MATLAB (LECO) |                        | -    |
|                 |                                        | ChromSquare       | Java (Shimadzu) |                        | -    |
|                 |                                        | ChromSpace®       | .NET (SepSolve Analytical) |                        | -    |
|                 |                                        | GasPedal          | Java (Decodon) |                        | -    |
|                 |                                        | AnalyserPro®      | C# XD (SpectralWorks) |                        | -    |
|                 |                                        | GC×GC-Analyzer™   | MATLAB (MsMetrix) |                        | -    |
| Published       | Peak deconvolution and detection        | PARAFAC and PARAFAC2 | MATLAB | PARAFAC2 peak detection robust to shifts in retention time by relaxing trilinearity rule. | 1, 2 |
|                 | Feature extraction / Data reduction     | INCA              | R (GC Image) | Normalisation and then alignment based on criteria for refining peak list. Used peak list from ChromaTOF. | 3    |
|                 | Feature extraction / Data ratio         | Tile-based Fisher | MATLAB | In-house tile-based F-ratio software. Supervised sample classification based on the Fisher ratios of binned regions of chromatogram described as 'tiles'. | 4    |
|                 | Peak detection / Feature extraction     | T-SEN              | R (GC Image) | Tool for targeted analysis and screening. Worked well on GC×GC coupled with high resolution mass spectrometry data. | 5    |
|                 | Feature extraction / Data pixel reduction | Discriminant     | MATLAB and C (GC Image) | Alignment based on dynamic time warping. Similar reduction method to Tile-based Fisher Ratio approach, using ANOVA and correlation significance testing on a pixel level. | 6    |
|                 | Classification                          | NMF                | R (GC Image) | Uses NMF R package with in-house R program for masking, resolution control and normalization. Performs non-negative matrix factorization for sample classification. Pre-processing included using local alignment plug-in in GC Image. | 7    |
|                 | Peak deconvolution and detection        | MCR-ALS            | MATLAB | Multivariate curve resolution-alternating least squares analysis for handling main data challenges including baseline correction, retention time shifts, and deconvolution. | 8    |
| Feature extraction / Data reduction | Method | Description |
|-----------------------------------|--------|-------------|
| NPLS | MATLAB | Uses 2D-asymmetric least squares algorithm for baseline correction prior to N-way partial least squares algorithm for data reduction. |
| DotMap | MATLAB | Pixel-based method based on mass spectral matching, used for targeted analysis and screening. Performed PARAFAC prior to analysis. |
| msPeak and msPeakG | R | Finds peak regions based on Normal-Exponential-Bernoulli and in later version by Normal-Gamma-Bernoulli models and then detects peaks within region based on probability models. |
| JS Divergence | MATLAB | Implemented through a GUI. Calculates dissimilarity metric between local regions of chromatogram and then probability of the regions being significantly different. |
| Cylindrical mapping | MATLAB and SAS | Addresses wrap-around phenomenon using a cylindrical transformation. Aligns peaks using a similarity metric based on cylindrical distance and mass spectral correlation. Baseline correction, peak finding and library matching performed in ChromaTOF. |
| Sibson-natural neighbour | MATLAB | Alignment of local regions based on Voronoi decomposition around reference peaks to determine weighting followed by Sibson natural-neighbour interpolation. First baseline correction was performed in GC Image. |
| Piecewise alignment | MATLAB | Alignment of local regions based on user defined window lengths and then maximizes the correlation coefficient for the scalar shift. |
| Dynamic time-warping | MATLAB | Pixel-based alignment based on dynamic time warping algorithm. |
| GRAM | MATLAB | Generalised rank annihilation method for peak detection following 2-D alignment. |
| Model-based peak alignment | R | Alignment uses a set of metabolite ‘landmark’ peaks. Metabolite matching based on empirical Bayes model. Peak merging first performed in ChromaTOF. |
| 2D-COW | MATLAB | 2D-correlation optimized warping for alignment. Used ChromaTOF for data processing and peak matching. |
| DISCO and DISCO2 | MATLAB | Distance and spectrum correlation optimisation alignment algorithm. Used peak list from ChromaTOF. Assumes a set of metabolite ‘landmark’ peaks present in every sample. |
| R2DGC | R | Uses output from ChromaTOF. Can filter problematic ions, perform mass spectral library matching and alignment based on similarity of peaks in exported peak lists. |
| GC’MS | Web server | Web server. Peak detection, baseline correction and alignment. Imports netCDF. Data parsing based on R. |
| Workflow / metabolomics pipeline | MetPP          | MATLAB | Peak filtering, merging, retention index matching, alignment, normalisation and statistical tests. Used peak list from ChromaTOF. |
|----------------------------------|---------------|--------|---------------------------------------------------------------------------------------------------------------------------------|
| Feature extraction / Data reduction | OCTpy       | Python | Uses output from ChromaTOF Statistical Compare feature.                                                                                                                                   |
| Workflow / metabolomics pipeline | Guineu       | Java   | Java-based. Modular design for user development. Alignment, normalisation, artefact removal, retention indices and functional group identification. First produce peak list from ChromaTOF. |
| Workflow / metabolomics pipeline | RMet         | R      | Import CDF files, supports visualisation, peak detection using MCR-ALS model and multivariate analysis.                                                                                  |
| Workflow / metabolomics pipeline | RGCxGC       | R      | Import netCDF files. Signal pre-processing including baseline correction and smoothing, alignment using 2D-COW algorithm and multivariate analysis. |
Figure S1. Example of a multi-stage process (e.g. an alignment based on batch number) automated by interfacing commercial and open-source software.
EXAMPLE FOLDER INSTRUCTIONS

To help users with the basics of running their workflow steps through the command line interface, we recommend either running the example scripts in the example folders (at https://github.com/rcfgroup/gc-automation) or first converting a current data processing method (.method) into a command file. Further instructions for running the example scripts are provided on pages S-6 – S-13 of this Supporting Information, in the files and in the Github repository. Converting a current method can be executed through the software GUI and instructions and familiarization with the command and batch file format are provided on pages S-14 – S-17 of this Supporting Information. These instructions and examples allow the user to become familiar with the format and check for errors before moving onto more complex custom tasks. A summary step-by-step list is provided below, with further details of each step provided in the ‘Before you begin’ section.

Step-by-step check list

1. Install GC Image
2. Install ‘Natural Neighbour (NIES-EPFL)’ plug-in
3. Install Python
4. Install ‘click’ and ‘pandas’ packages
5. Install Notepad++
6. Download and extract ‘Example Folders’
7. Follow instructions on page S-10.

Before you begin

The commercial and free and open-source software (FOSS) for your own custom workflow would already be installed. However, for this specific example, the following steps should be followed to ensure the example folders work as described.

Installing commercial software

Ensure a copy of the commercial software (in this instance GC Image) is installed (the examples were generated in version 2.8r2). A free trial of the software is available on request from https://gcimage.com/gcxgc/trial.html.

Installing plug-ins

The examples describe interfacing a published alignment algorithm. This algorithm is freely available as a plug-in at http://gcimage.com/forum/viewtopic.php?f=5&t=104 via the website under Plug-ins, for version 2.6 or later, and must be installed in order for the second and third example folders to work. To access the plug-in, login to your (free) user account. The original Matlab tool is available at https://github.com/jsarey/GCxGC-alignment. Once the plug-in file has been downloaded and extracted to the GC Image program folder, open GC Image, go to ‘Tools’ in the menu bar and from the list select ‘Manage Plugins’. Click the ‘Import’ button and locate and import the plug-in file. The ‘Natural Neighbour (NIES-EPFL)’ plug-in should now appear in the list of imported plug-ins. Click ‘Configure’ and change the parameters to match those below. Click ‘OK’ and close the program.

Installing FOSS

Ensure a compatible version of the free or open-source programming software (in this instance Python) is installed along with the packages used in the script (in this instance the ‘click’ and ‘pandas’ packages). Python is freely available at https://www.python.org/downloads/. If using a computer operating Windows 10 or later, you can download the Python app from the Microsoft Store.
To check if Python has been installed, in Windows press the windows key (\+)
+ R and type ‘cmd’ (without “ marks) in the Run window. This opens the command prompt window. Type ‘python’ and press enter. If Python has been installed it will come up with the program details as shown in Figure S2.

![Image of Python installation in command prompt window](image)

Figure S2: Checking the Python installation in the command prompt window.

To install the necessary packages, check if the package management system ‘PIP’ is installed. To do this in Windows, press the windows key (\+) + R and type ‘cmd’ in the Run window. This opens the a command prompt window. Next type ‘pip -V’. If this returns an error, PIP has not been installed. To install PIP, go to https://bootstrap.pypa.io/ and download the ‘get-pip.py’ file. Save the file to a known location. In the command prompt window, type ‘cd’ followed by the location of the get-pip.py file (Figure S3). Next, type ‘python get-pip.py’ into the command prompt and PIP will be installed. (Figure S3).

![Image of PIP installation in command prompt window](image)

Figure S3: Installing PIP in the command prompt window.

PIP can now be used to install the necessary packages by typing ‘pip install click’ and then ‘pip install pandas’. Following each command, the package will be installed. Lastly, import the packages by first entering ‘python’, this will show the Python program details acknowledging that you are now in the python interpreter (Figure S4), and then type ‘import click’ and then ‘import pandas’. You can now close the command prompt window.

![Image of importing packages](image)

Figure S4: Importing packages for the external scripts used in the folders.

Viewing files
For those inexperienced with basic coding we recommend installing Notepad++ to view the command and batch files. This software is freely available at https://notepad-plus-plus.org/downloads/.

Example folders
Download and extract the ‘Example Folders’ zip file from https://github.com/rcfgroup/gc-automation.

Extract the zip file before using and ensure the ‘Example folders’ folder is extracted (or copied once extracted) to the ‘C:\temp’ location. If the ‘temp’ folder can not be located create a new folder.

The ‘Examples folder’ (now with a path at ‘C:\temp\Example folders’) contains three folders: 1 Match Template, 2 Export Match File and 3 Apply Match File. These three folders are examples to help a beginner become familiar with the concept of using the command line interface to integrate free and open-source software with commercial software for GC×GC data processing. The examples in the folders wouldn’t necessarily be used independently as part of a workflow but effectively demonstrate this new way of being able to process GCxGC data.

Please note, if using a different version of the commercial software (not v2.8r2), you may need to change the path for the Command Line. In the first and second example folders, this can be changed in the batch (.bat) file (Figure S5A). In the third example folder, this can be changed in the python (.py) file (line 41, keeping the double slash formatting). The command line interface can be found in the program directory. An example path is:

C:\GC Image\GC Image 2.8r2 GCxGC (64-bit)\bin\CommandLine.bat.
Right-click on the .bat file in the first or second example folder, or the .py file for the third example folder, and select ‘Edit with Notepad++’. In the .bat file, under ‘set GC_IMG=’ paste the new path and then save the .bat file (as in Figure S5A). In .py file, under line 41 after ‘cmd =’ enter the new path using the double slash format shown in Figure S5B.

(A)
```
rem set full path for GC Image CommandLine.bat file (change from C:\ onwards with own filepath if not the same)
set GC_IMG=C:\\GC Image\\GC Image 2.8r2 GCKGC (64-bit)\\bin\\CommandLine.bat
```
(B)
```
41 | | | cmd = "\\C:\\\GC Image\\GC Image 2.8r2 GCKGC (64-bit)\\bin\\CommandLine.bat" -sysu
```

Figure S5: Changing the path for the command line interface in (A) the batch file and (B) in the python file.

When running the batch files (as described on page S-10) using a different version of the commercial software, the following message may appear.

Click ‘OK’ and the analysis will continue. The message will appear three times for each file being processed, click ‘OK’ to continue. This is not an issue for completing the examples, however, for custom workflows with more files the version information should be changed in the command and batch files.
Explanation of example folders

The first folder (1 Match Template) is a simple exercise, telling the commercial software to perform two tasks using the command line interface. These steps can also be performed in the GUI. This example is to get the user familiar with the translation between the GUI and command and batch files.

The second folder (2 Export Match File) tells the commercial software to perform two simple tasks again (matching a template and exporting a summary report) but this time the batch file incorporates an external FOSS, in the form of a Python script. Once the commercial software completes the tasks, the Python script uses the outputs (template file and summary report) to automatically generate an exported match file. The generation of multiple match files is a task which can't be batch processed within the GUI.

The third folder (3 Apply Match File) uses the processed chromatograms (Output folder) from the first folder (1 Match Template) and the exported match files (Exported Match Files folder) from the second folder, to perform a more complex list of commands, aligning the chromatograms using the Gros et al local alignment algorithm (available as plug-in). This time the commercial software is run through the Python script (as opposed to the Python script appending to the end). The Python script performs an automatic match between the batch number in the exported match files and the batch number in the chromatogram filename. Once it has matched them, it uses the external algorithm automatically align each chromatogram using each batch's unique match file. This intelligent iterative matching can not be performed within the GUI.

The example data in the first and third folder are chromatograms of a simple reference mixture with small variation in secondary retention times between the samples, especially for compounds with higher \( t_R \) (e.g. aromatics). The files have three different batch numbers (180808 15, 180815 1, and 180815 9) representing samples run at different times. The example data in the second folder are chromatograms of an n-alkane and aromatics mixture run on the corresponding dates as the reference mixture with the same batch numbers. To save memory space and make the demonstration as simple as possible the chromatograms are GCxGC-FID files, which have been exported a model image and saved as .gci files.
Instructions

1 Match Template folder – Instructions

In the ‘Input’ folder there are three chromatograms (six files in total, three .gci and three .bin files) labelled ‘ref mix’ with three different batch numbers (180808 15, 180815 1, and 180815 9). The command line window will open showing each file being processed. The command window will then close and three processed chromatograms (six files in total, three .gci and three .bin files) will now have appeared in the Output folder. The number of chromatograms can be changed (e.g. >00s), and the processing time will change accordingly, however, this example highlights the simplicity of sharing and repeating a workflow and gets the user familiar with the interface. (see Results on page S10).

2 Export Match File folder - Instructions

In the ‘Input’ folder there are three chromatograms (six files in total, three .gci and three .bin files). To run the analysis simply double-click the .bat file. The command line window will open showing each file being processed. The command window will then close and three processed chromatograms (six files in total, three .gci and three .bin files) will now have appeared in the Output folder. In the ‘Exported Match Files’ folder three .csv files will now have appeared. These are the exported match files for each batch based on the change in retention positions of the n-alkanes and aromatic compounds in the mixture. (see Results on pages S10).

3 Apply Match File folder – Instructions

Firstly, copy the files of the three chromatograms of the reference mixture (.gci and .bin files) from the Output folder of the first example folder (1 Match Template), and paste them into the Input folder in this folder. Next, copy the three exported match files (.csv files) from the Exported Match Files folder of the second example folder (2 Export Match Files), and paste them into the Exported Match Files folder in this folder.

To run the analysis simply double-click the .bat file. The command line window will open showing each file being processed (Figure S6). The command window will then show a message saying ‘Process finished’ and ‘Press any button to continue’. Press the ‘enter’ button on the keyboard and the command line window will close. Three processed chromatograms (six files in total, three .gci and three .bin files) will now have appeared in the Output folder. In the ‘Exported CSV Files’ folder three .csv files will now have appeared. These are the chromatograms exported as a single-column vector .csv file. The processed chromatograms can be reviewed in Investigator (see Results section on pages S12 – S13).

Figure S6: Example of command line window showing stream of processed files
Results

1 Match Template folder - Results

The Input and Output files can be viewed in GC Image. Open GC Image and click the yellow folder icon in the top left corner (Figure S7). Navigate to the C:\temp\Example folders\1 Match Template\input folder, select a chromatogram to open. You will see a series of peaks, however, in the chromatograms from the Input folder the chromatograms will appear unprocessed without any peaks (or ‘blobs) detected. In the chromatograms from the Output folder the chromatograms will have the series of peaks detected (with yellow and red outlines around them) and the peaks above the hydrocarbon series with yellow outlines (the carbonyl, terpene and aromatic peaks) will have been matched to the template and have compounds names associated with them.

![Figure S7: Opening an example chromatogram in GC Image](image)

To access the processing steps that have been performed on the chromatograms, the user can open the command and batch files. To open the command file (.cmd) and the batch file, right-click on a file and select ‘Open with’ and then ‘Notepad’ or ‘Edit with Notepad++’.

2 Export Match File folder - Results

The chromatograms in the Input folder show the separation of a mixture of n-alkane and aromatic compounds separating across the 2D chromatographic space. The peaks were detected and a template applied labelling the peaks across all the samples. This can be done in the commercial software GUI or using the interface as demonstrated in the first example. The chromatograms in the Output folder are the same, only the alignment template, comprising a subset of the compounds, was applied to the peaks and a summary report of the peak positions generated as specified in the commands file. The Python script in the folder took the details of the reference positions of the subset of peaks from the template file, and the actual retention positions of the subset of peaks in the chromatogram from the summary report and combined them to reproduce match files (as can be produced one at a time in the GUI) for each of the chromatograms automatically. This example demonstrates how interfacing commercial software and free and open-source software can be used to automate custom iterative tasks.
3 Apply Match File folder - Results

The input and Output files i.e. the chromatograms pre- and post-alignment using the Gros et al local alignment algorithm, can be compared in Investigator (part of the GC Image suite).

To do this, firstly open Investigator. In the top left of Investigator go to ‘File’ and click ‘Load Images’. Navigate to the C:\temp\Example folders\3 Apply Match Files\Input folder, select all three chromatograms holding the shift key and click ‘Open’.

In the ‘Load Options’ window (see Figure S8), uncheck the ‘Use configuration’ box and in the ‘Features’ tab select the ‘As is’ radial button. Next, click on the ‘Attributes’ tab, select ‘Analyse specific blob/area attributes’ and select the ‘Choose retention and response attributes’ radial button. In the list only check the ‘Retention I’, ‘Retention II’ and ‘Volume’ boxes, and uncheck the ‘Analyse specific blob set attributes’ box (Figure S9). In the ‘Class Assignment’ tab, select all three files by clicking whilst holding the shift key, click ‘Assign Class Label’ and choose ‘New Label’. In the box enter ‘Pre’ or an equivalent label and then click ‘Ok’, and then click ‘Ok’ again in the bottom left hand corner on the window. Investigator will then load the chromatograms into the program.

![Figure S8: Loading the processed chromatograms in Investigator to compare the secondary retention times pre- and post-alignment](image)

Next, repeat the above to load the processed chromatograms, this time navigating to the C:\temp\Example folders\3 Apply Match Files\Output folder, selecting all three chromatograms holding the shift key and click ‘Open’. All the options in the ‘Load Options’ window will now be greyed out, except for the ‘Class Assignment’ tab; highlight the new files and label them ‘Post’ or an equivalent label.

All six chromatograms should now be loaded into Investigator. If available in your version of the software, click ‘Analysis’ in the menu bar at the top and select ‘Custom settings’. In the ‘Custom Analysis Settings’ window, only check the ‘Class Mean’, ‘Class Stdev’, ‘Class %RSD’ and ‘Pairwise Mean Difference’ boxes and click ‘Ok’. If ‘Analysis’ is not available, go to the next step.

Next, click the ‘Attributes’ tab (see Figure S9) and in the left-hand column select ‘Retention II’. In the table on the right-hand side it is now possible to compare the %RSD(Pre) and %RSD(Post) columns. If ‘Analysis’ was not available to define the %RSD columns, the comparison can still be made by scrolling across to see the RSD columns labelled ‘Pre’ and ‘Post’. For all the compounds, the variation in secondary retention time is reduced in the post-alignment chromatograms (Figure S9). This analysis was implemented by double-clicking a .bat file. For further information on using Investigator please refer to the manufacturer guidance.
In this example, the command file was run through the custom Python script, demonstrating how commercial and FOSS can be used to automate complex custom workflows. The external script instructed the software to find the date and tray number at the beginning of each chromatogram (e.g. a new function added by the user), and align the chromatogram (e.g. an existing function within the commercial software) using the algorithm developed by Gros et al (e.g. integrating published open-source code) based on the corresponding match files automatically generated (i.e. new custom function from previous example) from the n-alkane/aromatic reference sample with same date and tray number. In effect, alignment vectors were automatically exported and then used to align all the samples based on batch and tray number, correcting variation in retention positions captured by the n-alkane and aromatic reference mixture.

In this example, the reference mixture of terpenoids, aromatics and carbonyls represents samples, but for demonstration purposes using a reference mixture with known compounds (independent of the alignment mixture) helps demonstrate the workflow implemented through the interface.
FAMILIARISATION AND CREATION OF NEW INTERFACE FILES

The steps for integrating external scripts are described and examples that include external scripts (with example data) have been made available at [https://github.com/rcfgroup/gc-automation](https://github.com/rcfgroup/gc-automation). However, as the integration of external scripts is to increase flexibility, the final implementation is up to the user.

The steps described below are for using with GC Image v2.8 or later. A free trial of the software is available on request from [https://gcimage.com/gcxgc/trial.html](https://gcimage.com/gcxgc/trial.html). Similar methods can be developed using different software. An alpha-version of a Python interface has been developed that can be made compatible with any software; automatically generates the command and batch files and can be made available on request.

Locating the command line interface

The command line interface can be accessed through the program directory. The command line can be found in the program folder, an example file path is shown below:

```
C:\GC Image\GC Image 2.8r2 GCxGC (64-bit)\bin\CommandLine.bat
```

If multiple versions of the software are installed, locate the command line in the version you wish to use.

Creating a command file

The command file is the list of processing steps the user would like to perform on the data files (e.g. baseline correction, peak detection) based on the main functions in the commercial software. The command file can be created by directly exporting the method from the software. The individual processing steps can also be accessed directly from the program directory.

For beginners, we recommend using Notepad or Notepad++, a free user-friendly software for basic programming. A new command file can be created in a text file `.txt` and then saved as a `.process.cmd` file, or the example `.cmd` files provided in the github repository can be saved and edited in Notepad++.

The command file has the following structure:

- `Configure C:\temp\configuration file.cfg`
- `<script>`
  - `<cmd name = “command name” label = “label name”>`
  - `<timeStamp> date and time </timeStamp>`
  - `<username> local user </username>`
  - `<cmdversion> software version </cmdversion>`
  - `<do>`
    - `<parameter id = “parameter” label = “parameter label”> file path </parameter>`
  - `</do>`
  - `</cmd>`
- `</script>`

Below is an example of a command file containing the commands to detect peaks and then match them to a template copied directly from an exported method:

```
-Configure C:\temp\file path\configuration file.cfg
<script>

<cmd name = “cmdBlobDetection” label = “Blob Detection”>
<timeStamp> 11 March, 2019 9:34:0</timeStamp>
$username> localuser </username>
$cmdversion> 2.8.2 </cmdversion>
<do>
</do>
</cmd>

<cmd name = “methodCmdMatchTemplate” label = “Match Template”>
<timeStamp> 11 March, 2019 9:39:49 </timeStamp>
$username> localuser </username>
$cmdversion> 2.8.2 </cmdversion>
<do>
  <parameter id = “filename” label = “Template File”> C:\temp\filepath\template file.bt </parameter>
  <parameter id = “updateCount” label = “Transform Updates before Applying”> 0 </parameter>
  <parameter id = “matchType” label = “Match Type”> default </parameter>
  <parameter id = “blobSet” label = “Match Set”> </parameter>
</do>
</cmd>
</script>
```

## In this example you can see the two commands, the first for peak detection or 'Blob Detection' and the second for template matching or 'Match Template', between the `<cmd>` </cmd> tag operators, which are contained within the `<script>` </script> tag operators.

## The Match Template command requires additional parameters such as the `.bt` file, which is given between `<parameter>` </parameter> tag operators, the user can use any number of commands. These parameters mirror the options available in the software ‘Method Editor’ GUI (GC Image -> Method -> Manage & Run Methods -> Edit Method -> Apply Template option).
Create a folder, labelled with the name of the workflow or process, and save the command file and batch file within it. The folder needs to contain an input folder, output folder and any additional files used in the analysis. We recommend using the example folder trees in the Github repository provided.

An example of the folder tree for the example above, saved in the C:\temp folder, is shown below:

The folder contains an input folder, an output folder, the configuration file, the command file, the batch file and additional files that the method requires such as a template file.

Exporting a method from the software

First create a method and use it to analyse a sample chromatogram or set of chromatograms. A method can be created from scratch, see software manual for details:

GC Image
> Method
> New Method
> Set configuration file & method settings

To export a method directly from the software:

GC Image
> Method
> Save Script

Save the script as an .xml file.
After saving the script, locate the .xml:
Right-click .xml
> Open with
> Notepad
or
> Edit with Notepad++

Then copy all the commands, including the <cmd> </cmd> tag operators and all the information between the operators, to the command file between the <script> </script> tag operators.
Creating a batch file

The batch file (.bat) tells the command line interface to apply all the steps outlined in the command file to all the files (sample chromatograms, .gci in this case) in the input folder. In the batch file, the user specifies the file path for the command line interface, the location of the command file, and the paths for the input and output folders. The batch file also provides the user an opportunity to integrate their free and open-source code.

Once a command file and batch file have been created, to run the analysis on the files in the input folder, the user double clicks the batch file. The command line interface will then open showing the data files being processed.

The structure of a batch file is more flexible and user-specific. An example structure of a batch file is shown below (green = user notes, blue = setting directories, black = user input required for setting path). We recommend viewing an example batch file provided in the Github repository. The orange code is a loop command telling the software to apply the data processing steps outlined in the command file to all the files in the input folder and save the processed chromatograms in the output folder.

```
rem Usage
rem Lines start with rem are comments and are safe to delete
rem This code will loop through all images (*.gci) under directory set in APP_IN and process each one
rem Please change paths below to corresponding directories
rem All images to be processed must be under the directory specified in APP_IN
rem No double quotes required for path variables below

rem set full path for GC Image CommandLine.bat file
set GC_IMG=C:\GC Image\GC Image 2.8r2 GCxGC (64-bit)\bin\CommandLine.bat

rem set where process method cmd file is located
set APP_PROC=C:\temp\filepath\command file.process.cmd

rem set where images to be processed are located
set APP_IN=C:\temp\filepath\input

rem set where output images are going to be saved
set APP_OUT=C:\temp\filepath\Output

rem save current working directory
set CURRENT_DIR=%cd%

rem switch into input directory
cd "%APP_IN%"

rem loop through each *.gci file and process it
for %i in (*.gci) do "%GC_IMG%" -sysu -cmdFile "%APP_PROC%" -s "%APP_IN%\%i" -d "%APP_OUT%\%i" 2>&1 1>nul | more

rem switch back to saved working directory
cd "%CURRENT_DIR%"
```
Integrating external scripts

The command and batch files provide the user an opportunity to integrate external scripts developed in free and open source software (FOSS). The functionality of the external script is defined by the user dependent on the application and workflow.

An example of how a Python script has been incorporated within a batch file is shown below. We recommend viewing the example batch files with external FOSS scripts integrated via the Github repository.

```bash
rem Usage
rem This code will loop through all images (*.gci) under directory set in APP_IN and process each one
rem All images to be processed must be under the directory specified in APP_IN
rem No double quotes required for path variables below

set full path for GC Image CommandLine.bat file
set GC_IMG=C:\GC Image\GC Image 2.8r2 GCxGC (64-bit)\bin\CommandLine.bat

set where process method cmd file is located
set APP_PROC=C:\temp\filepath\command file.process.cmd

set where images to be processed are located
set APP_IN=C:\temp\filepath\Input

set where output images are going to be saved
set APP_OUT=C:\temp\filepath\Output

set PYTHON_SCRIPT=C:\temp\filepath\external python script.py
set TEMPLATE=C:\temp\filepath\template file.bt
set REPORT_SOURCE=C:\temp\filepath\Reports\Summary Reports's_xml_files
set TEMPLATE_MATCH_DESTINATION=C:\temp\filepath\Exported match

rem save current working directory
set CURRENT_DIR=%cd%

rem switch into input directory
cd "%APP_IN%"

rem loop through each *.gci file and process it
for %%i in (*.gci) do "%GC_IMG%" -sysu -cmdFile "%APP_PROC%" -s "%APP_IN%\%%i" -d "%APP_OUT%\%%i" 2>&1 1>nul | more python "%PYTHON_SCRIPT%" --template="%TEMPLATE%" --source="%REPORT_SOURCE%" --destination="%TEMPLATE_MATCH_DESTINATION%"

rem switch back to saved working directory
cd "%CURRENT_DIR%"
```

Additional directories for performing the new function have been added e.g. set PYTHON_SCRIPT=. The above example includes a command file which detects peaks and matches the peaks to a template, specified in the set TEMPLATE= path. The command file also generates a summary report which contains the retention time information for all the peaks detected, the path for which is given in the set REPORT_SOURCE= path. The new Python script combines the information from the template file with the information of the detected peaks in the summary report, to automatically generate a match file for every chromatogram in the input folder. The processed chromatogram is saved to the output folder, under set APP_OUT=, and the exported match file produced by the Python script is exported to the folder specified under set TEMPLATE_MATCH_DESTINATION=.
Accessing the command line in other software

As mentioned, this proof-in-principle is not limited to GC Image. To demonstrate how the command line can be used in other software we also include details of how to access it in another frequently used data processing software, AnalyzerPro (SpectralWorks).

In a similar fashion, a batch file can be created. This contains all the steps the user would like to perform on the file(s) such as loading a specific method, applying the method to the files and then exporting the results.

This batch file can be executed through the command line interface:

Open the command prompt

> Run path/AnalyzerPro.exe “path/BatchFile.xml”

An example batch file as a .xml file is given below.

```xml
<?xml version="1.0"?>
<analyzerpro_batch_file>
  <run>
    <data_filename type="raw">C:\Program Files\SpectralWorks\AnalyzerPro\Raw Data\Agilent.D\DATA.MS</data_filename>
    <method type="swm">C:\Program Files\SpectralWorks\AnalyzerPro\Methods\Method1.swm</method>
    <commands>
      <command>load_data</command>
      <command>load_method</command>
      <command>process</command>
      <command>export</command>
    </commands>
  </run>

  <run>
    <data_filename type="swx">C:\Program Files\SpectralWorks\AnalyzerPro\Data Files\steroids02.swx</data_filename>
    <method type="swm">C:\Program Files\SpectralWorks\AnalyzerPro\Methods\Method1.swm</method>
    <commands>
      <command>load_data</command>
      <command>load_method</command>
      <command>process</command>
      <command>export</command>
      <command>exit</command>
    </commands>
  </run>
</analyzerpro_batch_file>
```

## Software version.
## Load the batch file.
## Example using Agilent data.
## Select the data file(s) or format into an input folder.
## Specify relevant method.
## This is similar to setting the configuration file on page S-14.
## Process the data.
## Export the data.

## Example using native .swx file.

## In the ‘Process’ and ‘Export’ stages, custom functions could be integrated to automatically process them in FOSS, similar to the integration of Python script demonstrated on page S-17.
REFERENCES

1. Hoggard, J. C.; Synovec, R. E., Parallel Factor Analysis (PARAFAC) of Target Analytes in GC × GC–TOFMS Data: Automated Selection of a Model with an Appropriate Number of Factors. Anal. Chem. 2007, 79 (4), 1611-1619.

2. Skov, T.; Hoggard, J. C.; Bro, R.; Synovec, R. E., Handling within run retention time shifts in two-dimensional chromatography data using shift correction and modeling. J. Chromatogr. A 2009, 1216 (18), 4020-4029.

3. Almstetter, M. F.; Appel, I. J.; Gruber, M. A.; Lottaz, C.; Timischl, B.; Spang, R.; Dettmer, K.; Oefner, P. J., Integrative Normalization and Comparative Analysis for Metabolic Fingerprinting by Comprehensive Two-Dimensional Gas Chromatography–Time-of-Flight Mass Spectrometry. Anal. Chem. 2009, 81 (14), 5731-5739.

4. Parsons, B. A.; Marney, L. C.; Siegler, W. C.; Hoggard, J. C.; Wright, B. W.; Synovec, R. E., Tile-Based Fisher Ratio Analysis of Comprehensive Two-Dimensional Gas Chromatography Time-of-Flight Mass Spectrometry (GC × GC–TOFMS) Data Using a Null Distribution Approach. Anal. Chem. 2015, 87 (7), 3812-3819.

5. Zushi, Y.; Hashimoto, S.; Fushimi, A.; Takazawa, Y.; Tanabe, K.; Shibata, Y., Rapid automatic identification and quantification of compounds in complex matrices using comprehensive two-dimensional gas chromatography coupled to high resolution time-of-flight mass spectrometry with a peak sentinel tool. Analytica Chimica Acta 2013, 778, 54-62.

6. Vial, J.; Pezous, B.; Thiébaut, D.; Sassiat, P.; Teillet, B.; Cahours, X.; Rivals, I., The discriminant pixel approach: A new tool for the rational interpretation of GCxGC-MS chromatograms. Talanta 2011, 83 (4), 1295-1301.

7. Zushi, Y.; Hashimoto, S., Direct Classification of GC × GC-Analyzed Complex Mixtures Using Non-Negative Matrix Factorization-Based Feature Extraction. Anal. Chem. 2018, 90 (6), 3819-3825.

8. Parastar, H.; Radović, J. R.; Bayona, J. M.; Tauler, R., Solving chromatographic challenges in comprehensive two-dimensional gas chromatography–time-of-flight mass spectrometry using multivariate curve resolution–alternating least squares. Analytical and Bioanalytical Chemistry 2013, 405 (19), 6235-6249.

9. Mostafapour, S.; Parastar, H., N-way partial least squares with variable importance in projection combined to GC × GC-TOFMS as a reliable tool for toxicity identification of fresh and weathered crude oils. Analytical and Bioanalytical Chemistry 2015, 407 (1), 209-215.

10. Sinha, A. E.; Hope, J. L.; Prazen, B. J.; Nilsson, E. J.; Jack, R. M.; Synovec, R. E., Algorithm for locating analytes of interest based on mass spectral similarity in GC × GC–TOF-MS data: analysis of metabolites in human infant urine. J. Chromatogr. A 2004, 1058 (1), 209-215.

11. Kim, S.; Ouyang, M.; Jeong, J.; Shen, C.; Zhang, X., A new method of peak detection for analysis of comprehensive two-dimensional gas chromatography mass spectrometry data. The annals of applied statistics 2014, 8 (2), 1209-1231.

12. Kim, S.; Jang, H.; Koo, I.; Lee, J.; Zhang, X., Normal–Gamma–Bernoulli peak detection for analysis of comprehensive two-dimensional gas chromatography mass spectrometry data. Computational Statistics & Data Analysis 2017, 105, 96-111.

13. Barcaru, A.; Vivó-Truyols, G., Use of Bayesian Statistics for Pairwise Comparison of Megavariate Data Sets: Extracting Meaningful Differences between GCxGC-MS Chromatograms Using Jensen–Shannon Divergence. Anal. Chem. 2016, 88 (4), 2096-2104.

14. Weusten, J. J. A. M.; Derks, E. P. A.; Mommers, J. H. M.; van der Wal, S., Alignment and clustering strategies for GC×GC–MS features using a cylindrical mapping. Analytica Chimica Acta 2012, 726, 9-21.

15. Gros, J.; Nabi, D.; Dimitriou-Christidis, P.; Rutler, R.; Arey, J. S., Robust Algorithm for Aligning Two-Dimensional Chromatograms. Anal. Chem. 2012, 84 (21), 9033-9040.

16. Pierce, K. M.; Wood, L. F.; Wright, B. W.; Synovec, R. E., A Comprehensive Two-Dimensional Retention Time Alignment Algorithm To Enhance Chemometric Analysis of Comprehensive Two-Dimensional Separation Data. Anal. Chem. 2005, 77 (23), 7735-7743.

17. Vial, J.; Noçairi, H.; Sassiat, P.; Mallipatu, S.; Cognon, G.; Thiébaut, D.; Teillet, B.; Rutledge, D. N., Combination of dynamic time warping and multivariate analysis for the comparison of comprehensive two-dimensional gas chromatograms: Application to plant extracts. J. Chromatogr. A 2009, 1216 (14), 2866-2872.

18. Fraga, C. G.; Prazen, B. J.; Synovec, R. E., Objective Data Alignment and Chemometric Analysis of Comprehensive Two-Dimensional Separations with Run-to-Run Peak Shifting on Both Dimensions. Anal. Chem. 2001, 73 (24), 5833-5840.

19. Jeong, J.; Shi, X.; Zhang, X.; Kim, S.; Shen, C., Model-based peak alignment of metabolomic profiling from comprehensive two-dimensional gas chromatography mass spectrometry. BMC Bioinformatics 2012, 13 (1), 27.
20. Zhang, D.; Huang, X.; Regnier, F. E.; Zhang, M., Two-dimensional correlation optimized warping algorithm for aligning GCxGC-MS data. Anal. Chem. 2008, 80.
21. Wang, B.; Fang, A.; Heim, J.; Bogdanov, B.; Pugh, S.; Libardoni, M.; Zhang, X., DISCO: distance and spectrum correlation optimization alignment for two dimensional gas chromatography time-of-flight mass spectrometry-based metabolomics. Anal. Chem. 2010, 82.
22. Ramaker, R. C.; Gordon, E. R.; Cooper, S. J., R2DGC: threshold-free peak alignment and identification for 2D gas chromatography-mass spectrometry in R. Bioinformatics (Oxford, England) 2018, 34 (10), 1789-1791.
23. Tian, T.-F.; Wang, S.-Y.; Kuo, T.-C.; Tan, C.-E.; Chen, G.-Y.; Kuo, C.-H.; Chen, C.-H. S.; Chan, C.-C.; Lin, O. A.; Tseng, Y. J., Web Server for Peak Detection, Baseline Correction, and Alignment in Two-Dimensional Gas Chromatography Mass Spectrometry-Based Metabolomics Data. Anal. Chem. 2016, 88 (21), 10395-10403.
24. Wei, X.; Shi, X.; Koo, I.; Kim, S.; Schmidt, R. H.; Arteel, G. E.; Watson, W. H.; McClain, C.; Zhang, X., MetPP: a computational platform for comprehensive two-dimensional gas chromatography time-of-flight mass spectrometry-based metabolomics. Bioinformatics 2013, 29 (14), 1786-1792.
25. Titaley, I. A.; Ogba, O. M.; Chibwe, L.; Hoh, E.; Cheong, P. H. Y.; Simonich, S. L. M., Automating data analysis for two-dimensional gas chromatography/time-of-flight mass spectrometry non-targeted analysis of comparative samples. J. Chromatogr. A 2018, 1541, 57-62.
26. Castillo, S.; Mattila, I.; Miettinen, J.; Orešič, M.; Hyötyläinen, T., Data Analysis Tool for Comprehensive Two-Dimensional Gas Chromatography/Time-of-Flight Mass Spectrometry. Anal. Chem. 2011, 83 (8), 3058-3067.
27. Moayedpour, S.; Parastar, H., RMet: An automated R based software for analyzing GC-MS and GC×GC-MS untargeted metabolomic data. Chemometrics and Intelligent Laboratory Systems 2019, 194, 103866.
28. Quiroz-Moreno, C.; Furlan, M. F.; Belinato, J. R.; Augusto, F.; Alexandrino, G. L.; Mogollón, N. G. S., RGCxGC toolbox: An R-package for data processing in comprehensive two-dimensional gas chromatography-mass spectrometry. Microchemical Journal 2020, 156, 104830.