The Modular Application Toolkit for Chromatography-Mass Spectrometry

User Guide

Version 1.3-SNAPSHOT

Nils Hoffmann

September 22, 2014

Preface

Intended Audience

This guide is aimed at novice as well as at experienced users of the Modular Application Toolkit for Chromatography-Mass Spectrometry (Maltcms) wishing to get a step-by-step introduction to setting up and running processing workflows for raw data from chromatography-mass spectrometry experiments.

Additionally, this guide provides programmers and power-users with the required knowledge for extending Maltcms with custom functionality.

License agreement

Availability

The Maltcms distribution is available from maltcms.sf.net.
A selection of raw sample data, required for the instructional examples, is available here.

Acknowledgements

We would like to thank the developers and testers of Maltcms, namely Sören Müller, Rolf Hilker, Mathias Wilhelm and Kai-Bernd Stadermann for their contributions.

We would further like to thank Professor Dr. Jens Stoye of the Genome Informatics Group and Professor Dr. Karsten Niehaus of the Metabolome and Proteome Group at Bielefeld University for their expert advice and continued support.

```
        System.out.println("Workflow using commands " + ↓
↪ dw.getCommandSequence().getCommands());
        System.out.println("Workflow using inputFiles " + ↓
↪ dw.getCommandSequence().getInput());
        System.out.println("Workflow using outputDirectory " + ↓
↪ dw.getOutputDirectory());
        return dw;
    }

    /**
     *
     * @param args
     */
    public static void main(String[] args) {
        //Download the test files from
        //http://sf.net/projects/maltcms/files/maltcms/example-data/
        List<File> inputFiles = Arrays.asList(
                new File("glucoseA.cdf"),
                new File("glucoseB.cdf"),
                new File("mannitolA.cdf"),
                new File("mannitolB.cdf"));
        List<IFragmentCommand> cmds = Arrays.asList(new IFragmentCommand[]{
            new DefaultVarLoader(),
            new DenseArrayProducer(),
            new TICPeakFinder(),
            new PeakCliqueAlignment()});
        IWorkflow w = createWorkflow(
```

Listing 1: Programmatic assembly of a workflow

Contents

1. Introduction

1.1. Getting Started

- Please download and install Java Runtime Environment (JRE) or JDK version 1.7.0 or newer for your specific platform from http://www.oracle.com/technetwork/java/javase/downloads/index.html.
- Then Download Maltcms from http://maltcms.de and follow the installation instructions.

1.2. Installation Instructions

Maltcms currently has no automatic installation script. In order to set everything up correctly, please follow the instructions for your operating system.

1.2.1. Unix/Linux/MacOSX

Extract the downloaded zip archive to a directory of your choice. In order to have maltcms.sh available on your command prompt, set the following in your $HOME/.profileor$HOME/.bashrc (or in any other appropriate shell configuration file).

```
export PATH=$PATH:/PATH/TO/EXTRACTED/ARCHIVE/bin
```

Maltcms can be launched via the maltcms.sh script in its home bin directory. This will use the java executable available via the environment variable JAVA_HOME. Otherwise, the script will try to find the location of your java installation by checking, if the 'java' command is available on your path.

The scripts will prompt for the installation directory of Maltcms, if the location of maltcms.jar can not be determined by the script automatically, it will check the environment variable MALTCMS_HOME. If that does not exist, the script will prompt you for the explicit location.

If you would like to install the maltcms.sh script and other scripts apart from the maltcms installation location, we suggest to add

```
export MALTCMS_HOME=/path/to/your/maltcms/
```

to your $HOME/.bashrc or $HOME/.profile file. Alternatively, simply overwrite the environment variable within maltcms.sh to set it to a fixed location. For backwards compatibility, MALTCMSDIR is still supported, but maltcms.sh will print a deprecation warning!

1.2.2. Windows

Extract the downloaded zip archive to a directory of your choice. In order to have maltcms.bat available on your command prompt, please follow the instructions appropriate for your installed operating system. You may need system administrator privileges to change system variables!

Windows 7 / 8

Select 'Computer' from the Start menu and choose 'System Properties' from the top level actions bar or from the context menu. Click on 'Advanced system settings' and select the 'Advanced' tab in the dialog. Click on 'Environment Variables' and under 'System Variables' or 'User Variables', locate the 'PATH' variable, and double-click to edit it. If no variable named 'PATH' is available in the 'User Variables' section, select new, name the variable 'PATH'.

Windows Vista

Right click on your 'My Computer' icon on the desktop and select 'Properties' from the popup-menu. Click on 'Advanced system settings' and select the 'Advanced' tab in the dialog. Click on 'Environment Variables' and under 'System Variables' or 'User Variables', locate the 'PATH' variable, and double-click to edit it. If now variable named 'PATH' is available in the 'User Variables' section, select new, name the variable 'PATH'.

Windows XP

Select 'Start' and click 'Control Panel'. Then go to 'System' and click on 'Advanced'. Click on 'Environment Variables' and under 'System Variables' or 'User Variables', locate the 'PATH' variable, and double-click to edit it. If now variable named 'PATH' is available in the 'User Variables' section, select new, name the variable 'PATH'.

1.2.3. Editing the Path Variable

Add the absolute path of the Maltcms directory to the end of the 'Variable value' like so: Old:

```
C:\Program Files;C:\Winnt;C:\Winnt\System32
```

New:

```
C:\Program Files;C:\Winnt;C:\Winnt\System32;C:\Path\To\Maltcms\bin
```

Finally, open a new command prompt by typing the 'Windows' or 'Meta' key on your keyboard. Then type 'cmd.com' and hit enter. A new command prompt will open. Type 'maltcms', hit 'Enter' and you should see maltcms' help appear. It may be necessary to log out and back in before the changed settings take effect.

The script will figure out its installation path from the call automatically. It will prompt for the installation directory of maltcms, if you call Maltcms from outside the installation directory and the environment variable MALTCMS_HOME is not set. We recommend to add MALTCMS_HOME as an additional 'System Variable', by defining a new variable in the 'Environment Variables' dialog under 'User variables' with the name MALTCMS_HOME and the absolute path of the Maltcms installation directory as its value.

1.3. Running Maltcms

The following section requires a command prompt (shell), indicated by the prefixed '>', which should not be typed into the prompt. Additionally, this section requires that you have installed Maltcms and that it is available from your path.

If you are using Windows, please replace 'maltcms.sh' with 'maltcms.bat'.

1.3.1. Running ChromA/ChromA4D

Pipeline location: As of Maltcms 1.2.1, pipeline *.properties files have been relocated to cfg/pipelines and have been renamed to .mpl (Maltcms PipeLine) to better distinguish them from standard configuration files. However, the old *.properties will still work without change. You may consider to rename them for consistency though.

ChromA

To execute the standard pipeline for GC- and LC-MS, use

```
>maltcms.sh -c cfg/pipelines/chroma.mpl -i <INPUTDIR> -o <OUTPUTDIR> -f <FILES>
```

ChromA4D

To execute the available pipelines for GCxGC-MS, use

```
>maltcms.sh -c cfg/pipelines/chroma4D.mpl -i <INPUTDIR> -o <OUTPUTDIR> -f ↓
  ↪ <FILES>
```

If you do not supply any arguments, Maltcms will print all available arguments with a short explanation.

```
>maltcms.sh -- -h
```

prints command-line options for the script with explanations.

1.3.2. Advanced Usage

You can start the 32 bit commandline version by typing

```
>maltcms.sh
```

within the bin dir below your Maltcms installation directory, which will print out the input options that you can supply.

Alternatively, on a 64 bit system and with 64 bit VM you can call

```
>maltcms.sh -d64 ...
```

Sometimes, the default amount of memory used by the JAVA runtime environment (VM) is not sufficient. You can then call

```
>maltcms.sh -Xms1G -Xmx2G ...
```

where -Xms1G sets the minimum amount of memory used by the VM to 1GByte of Ram and -Xmx2G sets the limit of the maximum amount of memory available to maltcms to 2GBytes. If you have more RAM installed, you can always increase the latter limit.

If you use java on a 32 bit architecture and a 32 bit VM, more than 2G might not be available. Using java on 64 bit architectures and with a 64 bit VM allows larger amounts of memory to be allocated.

In order to define custom properties, you need to pass the -c argument to Maltcms:

```
>maltcms.sh ... -c myCustom.properties -i <INPUTDIR> -o <OUTPUTDIR> -f <FILES>
```

will use the file 'myCustom.properties' from the current directory. Please note that from version 1.2.1 onwards, Maltcms pipeline configurations are stored below cfg/pipelines and are distinguishable by the '.mpl' file extension (mpl -> Maltcms PipeLine). However, the new file extension is just a convention.

Recommended arguments for Maltcms are:

```
-i <INDIR> (e.g. /vol/data)
-o <OUTDIR> (e.g. /vol/output)
-f <FILES> (e.g. "*.cdf")
```

If no INDIR is given, FILES are resolved against the current working directory. If no OUTDIR is given, processing results will be created below the current working directory in a directory following the convention 'maltcmsOutput/USERNAME/DATETIME/'.

In order to recurse into the INDIR, you can add '-r' to the command line. As an alternative to defining the input directory explicitly and using a file glob expression (only *.EXT expressions will currently work), you can also define a comma separated list of absolute or relative file paths. Relative paths are resolved from the current working directory. Please note that paths containing spaces can only be handled, if the whole argument (all paths separated by commas) is flanked in '"'s.

You can alternatively omit INDIR, if you fully qualify the path before each file provided with -f. Theses files can also have a wildcard expression as their name, e.g. "*.cdf". You can use -f <FILES> multiple times to supply different base directories with different file wildcard expressions to Maltcms. If you want to include all files below those base directories matching the wildcard, you should also supply the '-r' option.

1.3.3. Other Features

Graphical Wizard UI

As of version 1.1 of Maltcms, the graphical wizard is no longer supported. We recommend to use Maui, the Maltcms User Interface instead. Please visit http://maltcms.de for more details on Maui.

Extending Maltcms with custom code

Maltcms allows to add functionality by adding custom jar files to the runtime classpath.

The classpath in maltcms.jar is created at build time and will not contain those jars, so you need to add them manually to the runtime classpath with the '-cp' option supplied to java, or by using the 'maltcms.sh' script, which will pick these up automatically.

Scripting with Groovy

To simply use Maltcms in a higher level workflow, you can easily write scripts in the Groovy programming language[1]. These scripts can be executed by calling

```
maltcms.sh -exec path/to/script.groovy <ARGS>
```

from the Maltcms installation base directory, where ARGS are optional parameters to your script.

[1]http://groovy.codehaus.org/

1.3.4. Input data formats

Currently, netCDF compatible input and output is supported, mzXML input works reliably for MS1 data, output is currently redirected to netCDF format. Additionally, mzData is supported as input format, but again, output is redirected to netCDF format. Recently, read support for mzML has been added via the jmzml library.

1.3.5. Configuration

To get some insight into the parameters used by Maltcms and possible alternatives, consult the properties files in cfg/, especially

- factory.properties
- graphics.properties
- io.properties

and for logging options

- log4j.properties
- logging.properties

Properties and settings for individual commands in a processing pipeline are located in the respective xml file below cfg/pipelines/xml/.

Custom configuration files can be provided to maltcms.sh/maltcms.bat, as well as to maltcms.jar using the -c option. Those custom options then override the default values. Additionally, java supports the setting of environment properties via the -DmyProp=myValue switch. These will always override properties with the same name in user- supplied and default configurations.

1.3.6. Controlled Vocabulary

TBD

1.3.7. Example data

Example data for maltcms in netcdf format along with alignment anchors can be downloaded individually from http://bibiserv.techfak.uni-bielefeld.de/chroma/download.html.

Alternatively, 1D and 2D chromatogram data can be downloaded from http://sourceforge.net/projects/maltcms/files/maltcms/example-data/.

Part I.

Using Maltcms

2. Installation

Maltcms provides a command line interface (CLI) for direct invocation from a system's command prompt. Maltcms is usually called by a platform dependent wrapper script that is located below Maltcms' installation location:

- Microsoft Windows: bin/maltcms.bat
- Mac OS X: bin/maltcms.command
- Linux/Unix: bin/maltcms.sh

These wrapper scripts provide additional options and comfort functions over the default Maltcms command line and are thus recommended for general use.

2.1. Maltcms Command Line Arguments

The following basic command line arguments apply to all operating system platforms:

```
-?                                    display this help
-a <file1,...>                        alignment anchor files (without ↓
  ↪ path prefix, assumes location as given with option -i)
-b,--createBeanXml <class1,class2, ...>   Creates a fragmentCommands.xml ↓
  ↪ file in spring format for all instances of ↓
  ↪ cross.commands.fragments.AFragmentCommand
-c <file>                             configuration file location
-e <file:///path1/,...>               extension locations (directories ↓
  ↪ or .jar files)
-f <file1,...>                        input files (wildcard, e.g. *.cdf, ↓
  ↪ file name only if -i is given, or full path)
-h                                    display this help
-i <dir>                              input base directory
-l <class1,...>                       print available service providers ↓
  ↪ (e.g. cross.commands.fragments.AFragmentCommand)
-o <dir>                              target directory for all output
-p                                    print resolved configuration
-r                                    recurse into input base directory
-s,--showProperties <class1,class2, ...>  Prints the properties available ↓
  ↪ for configuration of given classes
```

2.1.1. Handling Files

-r
-i
-f

2.1.2. Handling Output

Output Base-Directory Location

-o

Changing the Output Directory Hierarchy

-DomitUserTimePrefix
-Doutput.overwrite=true

Postprocessing of Workflow Results selecting and zipping processing results

2.2. Platform Dependent Wrapper Scripts

2.2.1. MS Windows

2.2.2. MacOS X

2.2.3. Linux/Windows

3. Usage Scenarios

3.1. One-Dimensional Chromatography

3.1.1. Pyrolysis–Gas Chromatography

3.1.2. Gas Chromatography–Mass Spectrometry

3.1.3. Liquid Chromatography–Mass Spectrometry

3.2. Two-Dimensional Chromatography

3.2.1. Modulation Methods

3.2.2. GCxGC–Mass Spectrometry

3.2.3. LCxLC–Mass Spectrometry

3.3. Putative Compound Identification

3.4. Quantification

Part II.

Developing Maltcms

4. Concepts

4.1. Testing Framework

Maltcms' testing infrastructure is based on JUnit 4.10. The maltcms-test-tools folder below the main Maltcms Maven project contains two essential sub-projects: maltcms-test-defaults and maltcms-test-data that are built and distributed individually from Maltcms. Both of these dependencies should be included on projects whishing to use the Maltcms testing infrastructure. The parent pom of Maltcms does this automatically for all its child projects.

4.1.1. Plain Unit Testing

Simple unit tests in Maltcms are easy to define, following the JUnit cookbook. First, create a new class below the src/test/java folder of your Maven project. Then, for each behaviour of your class that you want to test, create one declaratively named method, optionally with a test prefix to better distinguish it from auxiliary code. Add the @org.junit.Test annotation before or in the line above the method's signature. Within the method, implement your testing logic, using the usual Assert.assert* methods. An example is given in listing 4.1, assuming the class MyFirstClass is located in src/main/java/my/first/module. Run mvn test on your project and watch for the test summary output.

Listing 4.1: Example of a plain unit test

```
package my.first.module;

import org.junit.Test;
//static import since Java 1.5
import static junit.framework.Assert.*;

public class MyUnitTest {
        @Test
        public void testMyAssertion() {
                MyFirstClass mfc = new MyFirstClass();
                assertTrue(mfc.isReady());
        }
}
```

4.1.2. Integration Testing

The maven-failsafe-plugin provides the execution logic for integration tests. If you want to define your unit test to be an integration test, simply annotate your class with @Category(IntegrationTest where Category is in the package org.junit.experimental.categories and IntegrationTest is in the maltcms.test package.

```java
/**
 *
 * @author Nils Hoffmann
 */
@Slf4j
public abstract class AFragmentCommandTest {

    @Rule
    public TemporaryFolder tf = new TemporaryFolder();
    @Rule
    public SetupLogging sl = new SetupLogging();
    @Rule
    public LogMethodName lmn = new LogMethodName();

    protected void setLogLevelFor(String prefix, Level logLevel) {
        switch (logLevel.toInt()) {
            case Level.ALL_INT:
                sl.getConfig().put("log4j.category." + prefix, "ALL");
```

Listing 4.2: Abstract base test class showing logging configuration and annotation.

Integration tests are skipped by default, as they may be long-running and resource-consuming. To enable them, add -DskipITs=false to your Maven command line, e.g. mvn -DskipITs=false install. Details on the specific configuration for Maltcms can be found in Maltcms' pom.xml.

4.1.3. Using the Maltcms Testing Infrastructure

The module maltcms-test-defaults contains a number of convenience classes to make testing of Maltcms commands and application a lot easier. Within the package maltcms.test, there are currently three children. IntegrationTest is the marker interface used to identify tests that should be run by the maven-failsafe-plugin mentioned in section 4.1.2. SetupLogging is a TestWatcher that can be included in test classes in order to set up log4j compatible logging in conjunction with the slf4j binding for log4j. To do so, simply create a new member variable:

```java
@Rule
private final SetupLogging sl = new SetupLogging();
```

If you also use lombok, you can annotate your test case classes with the @Slf4j annotation and have a readily configured logging system available for your unit tests. To see this in action, have a look at the abstract test base classAFragmentCommandTest. A short excerpt is given here for reference:

Appendix

Listings