Integrating Your Own R Functions into the CAM-Java

The CAM-Java software can be easily expanded by adding new functions and scripts writing in R. And you can also make use of existing R functions, as well as the whole Java interface in the CAM-Java program when implementing new algorithms. Since our program contains two modules: the R module (core algorithms) and the Java module (GUI), this chapter will introduce you how to integrate new R functions or packages you want into both modules, and how will they be benefited from the existing features of the program.

## 1. integrating into the R Module

Adding your own R functions or scripts into the R module is quite straightforward. All R functions and scripts currently used in the CAM-Java are gathered in the folder "r\_func". So you can simply copy & paste your files in the same folder.

In the "r\_func" folder, it contains core functions (like CAM.R, CM.R, CAM-nWCA.R, etc.), as well as helper functions in separated subfolders like "/function/", "/function1/", etc. Each of these functions is designed to address a specific problem. For example, the "SL\_EM.R" is used specifically for solving the Expectation-Maximization (EM) problem. These functions could be convenient resources for you when you implementing your own algorithms. To use these functions, simply uses the “source()” command in R, like follows:

***source("CAM.R")***

***source("functions/SL\_EM.R")***

## 2. integrating into the Java Module

To integrate new R functions into the Java GUI interface could be a little tricky, yet still easy to perform. In the current version of the CAM-Java (version 04022013), one of the main features in the Java module is that the GUI can now automatically load existing algorithms in the program and let the user to choose from. In addition, it can also change its GUI layout to handle algorithm parameters settings for different algorithms. All these features can be gained without modifying a single line of Java code in the Java Module. You only need to do the following two steps, and then you can run your new R functions using the Java GUI.

### Step 1. Write a Main R Script;

First you need to write a small R script to connect your R functions and the Java GUI. This main script will be called every time you selected your algorithm to run in the Java GUI, and may contain only a few lines. Taking the file “Java-runCAM-nWCA.R” (the main R script for the algorithm “CAM-nWCA”) as an example, and you can see it does three things:

***source("CAM-nWCA.R")***

***CAM.result <- CAM.nWCA(t(X\_mask),K)***

Load and run core functions of your algorithm. Please note that you don’t even need to load input data and set function parameters, because they will be created automatically by the Java GUI interface.

***A\_est <- CAM.result[[1]]***

***S\_est <- CAM.result[[2]]***

***final.result <- list(Aest=A\_est)***

Create an R list object to store algorithm results, this will be passed to the Java GUI interface for displaying purpose. Note that you should ALWAYS name it as “final.result”.

***source("Java-saveData.R")***

***saveData(A\_est, t(S\_est))***

***detach()***

Save the results you are interested for future usage. We have provided a small convenient R script “Java-saveData.R” to save the results in CSV files, you can either use it or write your own results saving method.

As you can see, the main script for adding new algorithm could be quite simple. This is because our Java GUI can automatically perform a lot of tasks for you, as loading different types of data, setting parameters, and drawing figures during the calculation if needed. It knows everything it needed about your algorithm. This magic is achieved by writing an algorithm configuration file, as stated in step 2.

### Step 2. Write an Algorithm Configuration File

Algorithm configuration file is the key component in our Java module. The Java GUI uses them to acknowledge the information of all algorithms existing in the current CAM-Java program. This mechanism can significantly simplify the procedure of adding new algorithms, as the user don’t need to modify any source code of the Java GUI interface, all they need is to register their algorithm in an external configuration file.

The configuration file uses eXtensible Markup Language (XML) format, if you are not familiar with this format, please check the XML tutorial on the w3schools website:

<http://www.w3schools.com/xml>.

Generally speaking, the XML format is especially useful to represent hierarchy structures. It has two main components: ***Elements*** and ***Attributes***. All XML configuration files in our program are stored in the folder “r\_func/config”. As you can see, one XML file can contain algorithm configuration descriptions for one, or multiple algorithms. To add your own algorithm configuration block, you can either modify the existing XML file, or create a new one. Below is a general procedure you can follow to write it.

1. The main part of a XML file (XML body) may contains several configuration blocks. For each block it has the following elements hierarchy:

***<configuration>***

***<algorithm>***

***<parameter>***

***<figure />***

***</parameter>***

***</algorithm>***

***</configuration>***

Detailed descriptions of these elements are listed in table 1.

Table 1. Elements used in the XML configuration file of the CAM-Java

|  |  |
| --- | --- |
| **Element Name** | **Description** |
| configuration | The “configuration” element is the root element of each XML file. |
| algorithm | Each “configuration” element may have zero or more “algorithm” element. It represents one existing algorithm in the CAM-Java program. |
| parameter | Each “algorithm” element may have zero or more “parameter” element. It represents one function parameter of the algorithm. |
| figure | Each “parameter” element may have zero or more “figure” element. It represents one figure generated by the algorithm. The parent “parameter” element is used to determine whether to draw these figures. |

In order to further describe the algorithm, some elements may also have corresponding attributes. The complete lists of these attributes can be checked in table 2, 3, and 4, for element “algorithm”, “parameter”, and “figure” respectively.

Table 2. All available attributes for the element “algorithm” in the XML configuration file

|  |  |  |  |
| --- | --- | --- | --- |
| **Element “algorithm”** | | | |
| **Attribute Name** | **Description** | **Possible Values** | **Default Value** |
| name | The name of the algorithm. | Any string | “” |
| script | The filename of the main R script of the algorithm. | Any string | “” |

Table 3. All available attributes for the element “parameter” in the XML configuration file

|  |  |  |  |
| --- | --- | --- | --- |
| **Element “parameter”** | | | |
| **Attribute Name** | **Description** | **Possible Values** | **Default Value** |
| name | The name of the parameter. | Any string | “” |
| type | The data type of the parameter. | “Double|Integer|DoubleSet|  IntegerSet|Boolean” | “Double” |
| range | The possible value of the parameter. This attribute is only useful when the “type” is “DoubleSet” or “IntegerSet” | Any string | “” |
| default | The default value of the parameter. | Any string | “” |
| info | The description of the parameter. | Any string | “” |

Table 4. All available attributes for the element “figure” in the XML configuration file

|  |  |  |  |
| --- | --- | --- | --- |
| **Element “figure”** | | | |
| **Attribute Name** | **Description** | **Possible Values** | **Default Value** |
| useMainScript | Indicates whether the figure is generated by the main R script. | “true|false” | “true” |
| script | The filename of the additional figure drawing R script. This is only useful when the “useMainScript” is “false”. | Any string | “” |

1. When you finished writing the body of the XML configuration, don’t forget to add the following two lines as the XML header, if you have created a new file.

***<?xml version="1.0" encoding="UTF-8"?>***

***<!DOCTYPE configuration SYSTEM "config.dtd">***

1. Now you can save the file you modified, or put the new XML file into the “r\_func/config/” folder. After restarting the Java GUI, you will find your algorithm appear in the drop down list in the “Algorithm Setting” panel (figure 1). Also when you selected the new algorithm, the whole parameters setting panel will be changed automatically, based on your algorithm configuration. You can also find several examples of existing algorithms in this software under the “config/” folder. It is easy to create a new one by modifying the old XML files.

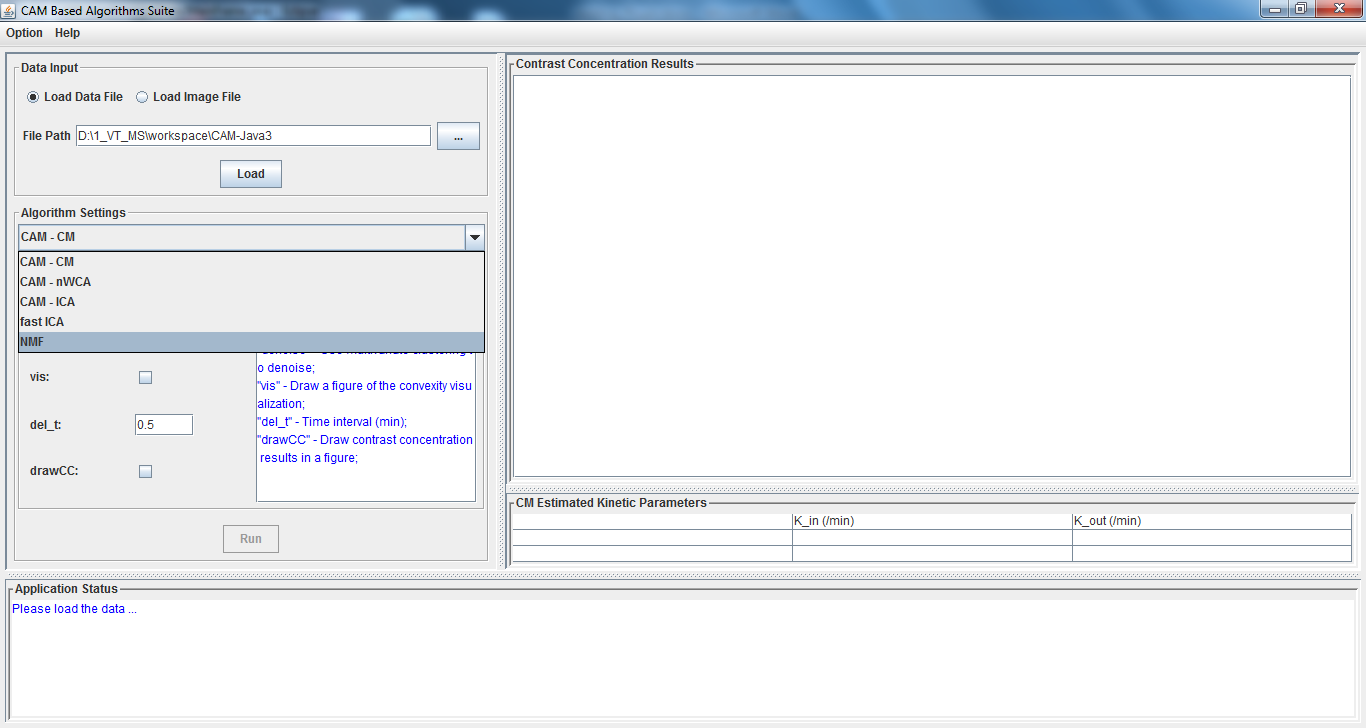


Figure 1. The algorithm selection drop down list in the Java GUI of the CAM-Java.

### Step 3. (Optional) Write an Algorithm Usage Demo.

This step is optional. You can also write a small demo R script to show how to use your new functions. This is a convenient way for other users to quickly understand how to use them directly in the R environment. You can put your demo file in the folder “r\_func/demo/”. Please note that these files are not used in the Java GUI interface.