Scripts

Local Image Features Extraction
— LIFEx —

C. Nioche, F. Orlhac, I. Buvat

LIFEx version 7.4.n,
Last update of document: 2023/03/19

List of Figures

Part I
What is a script?

Chapter 1
Introduction

1.1 Rationale and what is a script?

Scripting is the process of writing scripts that automate certain tasks within LIFEx application. Scripting is often used in this application to automate repetitive tasks, perform complex calculations, or customize the user experience.

In other words, a script file makes it possible to run all operations and calculations without any user interaction. You can prepare it in advance. You can have as many script files as you want, with names that you choose and you can save them and modify them. They are simple text files.

The scripting procedure for LIFEx application typically involves the following steps:

- Writing the script: you can start writing the script itself. This involves using the syntax and commands of the scripting language to create a set of instructions that the application can execute.

- Debugging and testing the script: After writing the script, you will need to test it to make sure it works as intended. This involves running the script and checking its output, and debugging any errors or issues that may arise.
- Integrating/Executing the script with the application: Once the script is working correctly, you will need to execute it with the application. This involves specifying when and how the script should be executed within the application.
- Maintaining the script: Finally, you will need to maintain the script over time to ensure that it continues to work correctly as the application evolves and changes. This may involve updating the script to work with new versions of the application, fixing any bugs that arise, or adding new functionality as needed.

1.2 What are the existing scripts for LIFEx?

- General-Script: these scripts allow you to perform mass image format changes (format conversion). For example, if you want to export series from DICOM format to nifti format. The details of all these scripts are written in part II.
- Texture-Script: this script allows to automate a texture analysis on a whole cohort of patients. The details of all these scripts are written in part III.
- MTV-Script: this script allows to automate an MTV analysis on a whole cohort of patients. The details of all these scripts are written in part IV.

1.3 General information about writing a script text file

Introduction: A script file for LIFEx is a text file containing a sequence of properties. Each property is written on a line.

Properties: Properties are configuration values managed as key=value pairs. In each pair, the key and value are both String values. The key identifies and is used to retrieve, the value, much as a variable name is used to retrieve the variable's value.

Property example: For example, an application capable of saving file might use a property named "$LIFEx.output.file$" to keep track of the directory used for the saving the file.

$$LIFEx.output.file = /home/user/results.csv$$

- key property is "$LIFEx.output.file$"
- value property is "$/home/user/results.csv$"

Construction guidelines of property:

- warning: a key is unique and cannot be repeated. Only one key per line.
- the order of the lines does not matter, however for human reading it is better to categorize the actions.
- file paths must always be written with a file separator "/" even under Windows.
- lines beginning with "#" are comments and are not interpreted by the application.

1.4 Script execution sequence

Here is the (mandatory) tasks order of execution in a script. Each script file will be read in full and executed as follows:

1. Series loading [mandatory]
2. — Series operations [optional]
3. — Series saving [optional]
4. — ROI loading [optional]
5. —— ROI operations [mandatory if "loaded ROI", otherwise optional]
6. —— ROI saving [mandatory if "loaded ROI", otherwise optional]
7. —— ROI extract features [optional and automatic]
8. —— ROI save result files [optional and automatic]
9. — ROI close [optional and automatic]
10. Series close [automatic]

1.5 How to run a script file

Before running a script:

1. Please check that you have no session file in the destination directory that you have specified in your script file;
2. The result filenames should NOT be `*TextureSession.csv`

Running the script :

- Please drag the script file in the panel used to load patient images.
- Alternatively, you can use the "browser + LocalDisk" interface to find and load your script file from your local disk.

1.6 Can I run several script files at the same time?

You can indeed run several script files one after another automatically. To do so, please select all script files and drag them in the panel used to put the patient image files. The script files will be executed one after another.

Beware: you can not load your script files using the "browser" interface, which supports only one script file at a time.

Part II
General concepts

Chapter 1
How to create a script?

1.1 Write a script

It is possible to chain the reading of the images and their respective ROI in order to obtain a including all results (.csv).

1.2 Minimal script

This section shows an script example. This script loads 2 image series named PT0 and PT1, and then 2 ROI associated with PT0 (in Session 0) and 2 ROI associated with PT1 (in Session 1).

You can copy/paste this script into a text file named script.txt (for example). Don't forget to change some information: ex. directory/subDirectory

```
########################################################################
```

```
# Common
#################################################################################
# fixed information on script -> mandatory
LIFEx.script = Main
LIFEx.script.version = 2.0

# result file -> mandatory
LIFEx.output.file={directory/subDirectory}/results.csv


#################################################################################
# Session0 / Series
#################################################################################
# loading series -> mandatory
LIFEx.Session0.Img0={directory/subDirectory}/PT0


#################################################################################
# Session1 / Series
#################################################################################
# loading series -> mandatory
LIFEx.Session1.Img0={directory/subDirectory}/PT1


#################################################################################
# Session 0 / ROI
#################################################################################
# loading ROI -> mandatory
LIFEx.Session0.Roi0={directory/subDirectory}/RoiVolume/R1.uint16.nii.gz
LIFEx.Session0.Roi1={directory/subDirectory}/RoiVolume/R2.uint16.nii.gz


#################################################################################
# Session 1 / ROI
#################################################################################
# loading ROI -> mandatory
LIFEx.Session1.Roi0={directory/subDirectory}/RoiVolume/R1.uint16.nii.gz
LIFEx.Session1.Roi1={directory/subDirectory}/RoiVolume/R2.uint16.nii.gz
```


1.3 Complete script: save anonymous DICOM Series from DICOM Series

See below for a complete script example of save anonymous DICOM Series from DICOM Series:

```
#################################################################################
# Common
#################################################################################
# fixed information on script -> mandatory
LIFEx.script = Main
```

```
LIFEx.script.version = 2.0

##########################################################################
# Session0 / Series
##########################################################################
# loading series -> mandatory
LIFEx.Session0.Img0={directory/subDirectory}/PT0
LIFEx.Session0.Img0.Operation0=Save anonymous series (.dcm)

##########################################################################
# Session1 / Series
##########################################################################
# loading series -> mandatory
LIFEx.Session1.Img0={directory/subDirectory}/PT1
LIFEx.Session1.Img0.Operation0=Save anonymous series (.dcm)
```


1.4 Main remarks relevant to the script writing

- What are the image formats that can be managed using LIFEx scripts?
  - image files can be in NIfTI-1 format (.nii or .nii.gz). In this case, the complete pathway to the patient image file should be given in the script;
  - to load DICOM images, you must define the root directory of these images (without the final filename). All files included in this directory will be loaded;
- What are the ROI formats that can be managed using LIFEx scripts?
  - the NIfTI-1 and RTStruct are supported in LIFEx scripts. In both cases, the file name should include the full path to the file. In the case of RTStuct, all ROI are loaded without exception.
- Syntax of all pathways of files in LIFEx scripts:
  - the syntax of pathway must neither contain accents and nor space.
  - the syntax of pathway must not necessarily conform to your system rules:
    - Windows: Unit:/Directory/File.extension or Unit:/Your Directory/ for series of DICOM images ; example C:/Home/Users1/File1
    - Linux: /Your Directory/Your File.extension
    - MacOs: /Your Directory/Your File.extension

Chapter 2
Perform operations

2.1 Introduction

Some intermediate operations are possible on the ROI. They will be performed after reading the Series and ROI but before extracting the features and recording the results:

Main syntax of one operation: The syntax property for one operation is:
{key}.Operation0=nameOp0,arg1,arg2,...,argn

with "nameOp0" is the title of the button of the ROI action to be performed, arg1 the first argument, arg2 the second argument, ...

Main syntax of many operations to the same Series or ROI: The syntax property for many operation is:
{key}.Operation0=nameOp0,arg1,arg2,...,argn|nameOp1,arg1,arg2,...,argn

with "nameOp0" is the first operation, "nameOp1" is the second operation.

On this line the order (from left to right) of the operations is respected during execution.

Order operations between lines: If you have more than one operation to perform (on a different series) you can describe the actions with an order given by the key_operationN:
{key}.Operation0=...
{key}.Operation1=...
{key}.Operation2=...

Perform operations

2.2 Series operations

2.2.1 Series operations

- **Syntax examples:**
  - Saving series with anonymous DICOM field:
    LIFEx.Session0.Img0.Operation0=Save anonymous series (.dcm)
  - Change unit of series in SUVbm:
    LIFEx.Session0.Img0.Operation0=SUVbw
  - 2x2x2 Resampling series then saving result series in nifti format:
    LIFEx.Session0.Img0.Operation0=resampling,2,2,2|save series (.nii uInt16)
  - Change output directory of series saving:
    LIFEx.Output.Directory = {directory/subDirectory/}

- **Available Series operations [file/edit menu]:**
  - Save anonymous series in DICOM format (only if loaded series is in DICOM format too):
    LIFEx.Session0.Img0.Operation0=**Save anonymous series (.dcm)**
  - Save series in DICOM format (only if loaded series is in DICOM format too):
    LIFEx.Session0.Img0.Operation0=**Save series (.dcm)**
  - Save series in nrrd format:
    LIFEx.Session0.Img0.Operation0=**Save series (.nrrd)**
  - Save series in nifti format (float 32 bits):
    LIFEx.Session0.Img0.Operation0=**Save series (.nii float32)**
  - Save series in nifti format (uint 16 bits):
    LIFEx.Session0.Img0.Operation0=**Save series (.nii uint16)**
  - Save MIP 3D Plans: Coronal, Sagittal, Axial (.nii float32):
    LIFEx.Session0.Img0.Operation0=**Save MIP 3D Plans**
  - Change output directory of Series saving:
    LIFEx.Session0.Img0.Operation0.OutputDirectory = {directory/subDirectory/}
  - Change common output directory of all Series saving:
    LIFEx.OutputDirectory = {directory/subDirectory/}
  - Unit of Y axis of series, choose between: kBq/mL || SUVbw || SUVlbm || SUVibw || SUVbsa || Cpx/vx || Gy/vx || %/vx || .# class || .# k Pa || HU || T || mL/100g || mL/100g/min || sec || RC (.###) || min-1 || Proba || # || #.# || #.## || #.### || #.####
    LIFEx.Session0.Img0.Operation0=**SUVbw**

2.3 ROI operations

2.3.1 ROI operations

- **Syntax examples:**
  - Below is an example of an "Absolute threshold" (n) with MinThreshold=2.5 and MaxThreshold=50:
    LIFEx.Session0.Roi0.Operation0=n,2.5,50
  - Below is an other example of a "Absolute threshold" (n) with MinThreshold=2.5 and MaxThreshold=50 followed by the saving of result ROI in nifti (.nii) format:
    LIFEx.Session0.Roi0.Operation0=n,2.5,50|nii Save

- **Available ROI operations [threshold menu]:**
  - Absolute threshold:
    LIFEx.Session0.Roi0.Operation0=**n,ValueOfMinThreshold,ValueOfMaxThreshold**
  - Percent threshold:
    LIFEx.Session0.Roi0.Operation0=**n%,ValueOfPercentThreshold**
  - 40 Percent threshold:
    LIFEx.Session0.Roi0.Operation0=**40%**
  - 70 Percent threshold:
    LIFEx.Session0.Roi0.Operation0=**70%**
  - Peak of 0.5cc:
    LIFEx.Session0.Roi0.Operation0=**Peak.5cc**
  - Peak of 1cc:
    LIFEx.Session0.Roi0.Operation0=**Peak1cc**
  - Nestle threshold:
    LIFEx.Session0.Roi0.Operation0=**Nestle,ValueOfThreshold**
  - PERCIST threshold (need to have previously loaded a ROI named Liver):
    LIFEx.Session0.Roi0.Operation0=**PERCIST**

- **Available ROI operations [file/edit menu]:**
  - Save ROI in nifti (.nii) format file:
    LIFEx.Session0.Roi0.Operation0=**nii Save**
  - Save ROI in DICOM (.dcm) format file:
    LIFEx.Session0.Roi0.Operation0=**dcm Save**
  - Save ROI in comma separator value (.csv) format file:
    LIFEx.Session0.Roi0.Operation0=**csv Save**
  - Save ROI in turkupetcentre (.dft) format file:
    LIFEx.Session0.Roi0.Operation0=**dft Save**
  - Not yet implemented (request of Recovery Coefficient not available in script):
    LIFEx.Session0.Roi0.Operation0=**dftrc Save**
  - Save ROI in Pmod (.nrdd) format file:
    LIFEx.Session0.Roi0.Operation0=**nrdd Save**
  - Change output directory of ROI saving:
    LIFEx.Session0.Roi0.Operation0.OutputDirectory = {directory/subDirectory/}
  - Change common output directory of all ROI saving:
    LIFEx.OutputDirectory = {directory/subDirectory/}

Part III
Texture-Script

Chapter 1
How to create?

Scripting procedure for texture calculation without user interaction

1.1 Rationale

When you have a large number of ROI and/or a large number of patients (`session` in script) for which you want to calculate usual indices (SUV, Volume, ...) and/or histogram-based or textural features, it can be convenient to run a script that will do all the calculations for you without any user interaction. ROI have to be prepared beforehand. You can then run the script many times if you want to perform the calculations using different sets of parameters.

For instance: You have 10 ROI per patient and a set of 100 patients to be processed. You are interested in studying the impact of the $nbGreyLevels$ parameter on the 10*100 ROI, by setting this parameter to 64, 128 and 256. You can create three script files that will all be identical except for the line that sets the $nbGreyLevels$ parameter. Running the script will calculate all indices for you automatically and store them in csv files. You will be able to run all three scripts using a single command line, see the "Can I run several script files at once?" p.12.

Another example: You have 10 ROI per patient and a set of 100 patients to be processed. You are interested in studying the impact of the $spatial\ resampling$ parameter on the 10*100 ROI, by setting this parameter to 2x2x2 mm and to 4x4x4 mm. You can write and run a first script by setting the $spatial\ resampling$ parameter to 2x2x2 mm. Then duplicate the script and change 2x2x2 mm by

4x4x4 mm to produce a second script to be run. Running the two scripts will calculate all indices for you automatically and store them in csv files. Setting the spatial resampling parameter to 2x2x2 mm in the script file can be done as follows:

```
# SpatialResampling of Img0 of session0
   LIFEx.texture.Session0.Img0.ZSpatialResampling=2
   LIFEx.texture.Session0.Img0.YSpatialResampling=2
   LIFEx.texture.Session0.Img0.XSpatialResampling=2
```

Using scripts may produce numerous ROIs. To check that the resulting ROI are compatible with texture index calculation and before running such calculations that can be time-consuming, you can use the CheckTex button after the ROI creation and before proceeding with textural index calculation. This will check whether the ROI includes a single cluster and contain a number of voxels greater than that required for consistent textural feature calculation. If one of these two conditions is not met, a warning message will be displayed.

1.2 How to write a script file?

The script file describes a list of functions to be executed within LIFEx.

- a) Reading the patient (series) file;
- b) Perform filter on series
- c) Reading one or several ROI files for that patient;
- d) Setting the parameters involved in the index calculation;
- e) Calculation of the different indices;
- f) Writing results in a new or existing csv file;
- g) Closing all files;
- h) Back to step a) if needed.

How to create?

Here is an example of script file based on an example provided with LIFEx. You can save it as `TextureScript.txt`. The lines starting with # are not interpreted by LIFEx, they only include comments.

```
LIFEx.MTV.script = Texture LIFEx.MTV.script.version = 2.0 ##########################################################
# filter (only one activate at same script) #
#############################################################################
#filter: Laplacian of Gaussian (LIFEx>=7.0.0) [ true || false ]
LIFEx.filter.LaplacianOfGaussian.Enable=false
#filter: DimensionProcessing= Laplacian of Gaussian [ 3D || 2D ]
LIFEx.filter.LaplacianOfGaussian.DimensionProcessing=3D
#filter: PaddingMethod Laplacian of Gaussian [ Reflect || Periodic || Edge || Zero ]
LIFEx.filter.LaplacianOfGaussian.PaddingMethod=Reflect
#filter: sigma Laplacian of Gaussian [ unit mm ]
LIFEx.filter.LaplacianOfGaussian.Sigma.z=7
LIFEx.filter.LaplacianOfGaussian.Sigma.y=2
LIFEx.filter.LaplacianOfGaussian.Sigma.x=2

#filter: Gaussian (LIFEx>=7.2.0) [ true || false ]
LIFEx.filter.Gaussian.Enable=false
#filter: DimensionProcessing= Gaussian [ 3D || 2D ]
LIFEx.filter.Gaussian.DimensionProcessing=3D
#filter: PaddingMethod Gaussian [ Reflect || Periodic || Edge || Zero ]
LIFEx.filter.Gaussian.PaddingMethod=Reflect
#filter: sigma Gaussian [ unit mm ]
```

```
LIFEx.filter.Gaussian.Sigma.z=7
LIFEx.filter.Gaussian.Sigma.y=2
LIFEx.filter.Gaussian.Sigma.x=2

#filter: Mean (LIFEx>=7.0.0) [ true || false ]
LIFEx.filter.Mean.Enable=false
#filter: Diameter Kernel Size (vx) [ integer value ]
LIFEx.filter.Mean.DiameterKernelSize=3
#filter: PaddingMethod [ Reflect || Periodic || Edge || Zero ]
LIFEx.filter.Mean.PaddingMethod=Reflect

#filter: Wavelet (LIFEx>=7.0.0) [ true || false ]
LIFEx.filter.Wavelet.Enable=false
#filter: Transform along Z axis(only for 3D) [ true || false ]
LIFEx.filter.Wavelet.ZTransform=true
#filter: Family [ Coiflets || Biorthogonal || Daubechies || Haar || Reverse biorthogonal || Symlets ]
LIFEx.filter.Wavelet.Family=Coiflets
#filter: Order [ integer value ] cf. documentation for values
LIFEx.filter.Wavelet.Order=1
#filter: Level [ constant integer value ] cf. documentation for values
LIFEx.filter.Wavelet.Level=1
#filter: PaddingMethod [ Reflect || Periodic || Edge || Zero ]
LIFEx.filter.Wavelet.PaddingMethod=Reflect

#filter: Laws (LIFEx>=7.0.0) [ true || false ]
LIFEx.filter.Laws.Enable=false
#filter: Kernel Size (vx) [ L3 || L5 || E3 || E5 || S3 || S5 || W5 || R5 ]
LIFEx.filter.Laws.ZKernelSize=L3
LIFEx.filter.Laws.YKernelSize=L3
LIFEx.filter.Laws.XKernelSize=L3
#filter: PaddingMethod [ Reflect || Periodic || Edge || Zero ]
LIFEx.filter.Laws.PaddingMethod=Reflect


##########################################################################
# texture #
##########################################################################
#texture: Common (LIFEx>=5.1.0)
LIFEx.texture.BinSize=0
LIFEx.texture.NbGrey=64
LIFEx.texture.SessionCsv=.../ScriptDemoMultiSeries/TextureResults.csv

#texture: Absolute (LIFEx>=5.1.0) [ true || false ]
LIFEx.texture.ButtonAbsolute=true
LIFEx.texture.MinBound=0
LIFEx.texture.MaxBound=20

#texture: RelativeMeanSd (LIFEx>=5.1.0) [ true || false ]
LIFEx.texture.ButtonRelativeMeanSd=false

#texture: RelativeMinMax (LIFEx>=5.1.0) [ true || false ]
LIFEx.texture.ButtonRelativeMinMax=false

#texture: DistanceWithNeighbours (LIFEx>=5.1.0)
LIFEx.texture.GLCM.DistanceWithNeighbours=1
```

```
#texture: dimension calculation (3D is default) (LIFEx>=5.1.0) [ 3D || 2D ]
LIFEx.texture.DimensionProcessing=3D

# Patient0
LIFEx.texture.Session0.Img0=.../ScriptDemoMultiSeries/PT0img.nii.gz
# SpatialResampling of Img0 of session0 (0 = no spatial resampling = native spacing voxels)
LIFEx.texture.Session0.Img0.ZSpatialResampling=0
LIFEx.texture.Session0.Img0.YSpatialResampling=0
LIFEx.texture.Session0.Img0.XSpatialResampling=0
LIFEx.texture.Session0.Roi0=.../ScriptDemoMultiSeries/R1.nii.gz
LIFEx.texture.Session0.Roi1=.../ScriptDemoMultiSeries/R2.nii.gz

LIFEx.texture.Session1.Img0=.../ScriptDemoMultiSeries/PT1img.nii.gz
# SpatialResampling of Img0 of session1 (0 = no spatial resampling = native spacing voxels)
LIFEx.texture.Session1.Img0.ZSpatialResampling=2
LIFEx.texture.Session1.Img0.YSpatialResampling=2
LIFEx.texture.Session1.Img0.XSpatialResampling=2
LIFEx.texture.Session1.Roi0=.../ScriptDemoMultiSeries/C1.nii.gz
```

Here are the functions that are executed when running this text file:

- Reading the images (`PT0img.nii.gz`) of patient0 (`=Session0`)
- No spatial resampling (because values=0)
- Reading `R1.nii.gz` ROI of patient0
- Reading `R2.nii.gz` ROI of patient0
- Reading the setting of the Common, Absolute, RelativeMeanSd, RelativeMinMax
- Calculating all texture features of `R1.nii.gz` and `R2.nii.gz` on `PT0img.nii.gz`
- Writing the results in an existing csv session file named `TextureResults.csv`
- Closing all ROI and patient0 data

then:

- Reading the images `PT1img.nii.gz` of patient1 (`=Session1`)
- Do a spatial resampling with 2x2x2 new spacing voxel
- Reading `C1.nii.gz` ROI of patient1
- Reading the setting of the Common, Absolute, RelativeMeanSd, RelativeMinMax
- Calculating all texture features on `C1.nii.gz` on `PT1img.nii.gz`
- Writing the results in an existing csv session file named `TextureResults.csv`
- Closing all ROI and patient1 data

1.3 Scripting grey-level discretization

1.3.1 How set the quantization

When you set the quantization of grey-level discretization parameters, you can leave one of the two sets to 0. Its value will then be automatically calculated based on the other parameter setting.

```
LIFEx.texture.BinSize=3.125
LIFEx.texture.NbGrey=128.0
```

For instance:
if $BinSize = 0$ then $BinSize = (boundMax - boundMin)/(nbGreyLevels - 1)$
if $nbGreyLevels$ = 0 then $nbGreyLevels = ((boundMax - boundMin)/binSize) + 1$
if $BinSize$ = 0 and $nbGreyLevels = 0$ then $nbGreyLevels = 64$ by default
$boundMin$ and $bounsMax$ are the minimum and maximum values in the processed ROI.

Discretization has been set with:

$$discretizedValue = floor((nbGreyLevels * \frac{originalValue - boundMin}{boundMax - boundMin)} + 1)$$

1.3.2 Have the minimum and maximum bounds of a whole cohort of patients in a script (without texture calculation)

It is easy to find the BoundMax of all the ROI of all your patients in order to put this bound in the final script. This has to be done in 2 steps with exactly the same script (by changing one line). Here is the process.

- step 1: add this line in the script: "LIFEx.check=true". It allows calculating the min, max bounds of all the ROI in a single pass of the script.
- step 2: run the script a first time to have only bound results; You then retrieve the max of the max of all the ROIs and correct the value in the script: LIFEx.texture.MaxBound=found value
- step 3: change LIFEx.check=false in place of LIFEx.check=true in the script file
- step 4: run the script a second time to have all feature results

1.4 Scripting spatial resampling

When you set the spatial resampling parameters, you can leave the three parameters $ZSpatialResampling$, $YSpatialResampling$, $XSpatialResampling$ set to 0 (or delete rows). This is equivalent to performing the calculation using the original voxel size of the images.

If you want to change the voxel size of the image, set the parameters to a value expressed in millimeters. The three mandatory values ($ZSpatialResampling$, $YSpatialResampling$, $XSpatialResampling$) can be different.

Part IV
MTV-Script

Chapter 1
How to create a MTV script

1.1 Introduction

The extraction of MTV values can be automated with scripts. It is possible to chain the reading of the images and their respective ROI in order to obtain a file including all results (.csv).

1.2 Script example

This section shows an script example of MTV protocol.

You can copy/paste this script into a text file named script.txt (for example). Don't forget to change some information: ex. directory/subDirectory

```
##########################################################################
# Common
##########################################################################
# fixed information on script -> mandatory
LIFEx.script = MTV
LIFEx.script.version = 2.0

LIFEx.check=false
LIFEx.keepOneLargest=false

# result file -> mandatory
LIFEx.output.file={directory/subDirectory}/MTV_results.csv


##########################################################################
# Session0 / Series
##########################################################################
# loading series -> mandatory
LIFEx.Session0.Img0={directory/subDirectory}/PT0


##########################################################################
# Session1 / Series
##########################################################################
# loading series -> mandatory
LIFEx.Session1.Img0={directory/subDirectory}/PT1


##########################################################################
# Session 0 / ROI
##########################################################################
# loading ROI -> mandatory
LIFEx.Session0.Roi0={directory/subDirectory}/RoiVolume/R1.uint16.nii.gz
LIFEx.Session0.Roi1={directory/subDirectory}/RoiVolume/R2.uint16.nii.gz


##########################################################################
# Session 1 / ROI
##########################################################################
# loading ROI -> mandatory
LIFEx.Session1.Roi0={directory/subDirectory}/RoiVolume/R1.uint16.nii.gz
LIFEx.Session1.Roi1={directory/subDirectory}/RoiVolume/R2.uint16.nii.gz
```


1.3 Output files

The root value of the key *"LIFEx.MTV.output.file"* will be used for the construction of 2 files (_ROI and _SUMMARY) will be saved at the end of the script execution by the application:

- MTV_results_ROI.csv will contain all the features extracted from each ROI;
- MTV_results_SUMMARY.csv: will contain aggregated features calculated from several ROIs.