



AUTOMATING THE HANDLING OF MULTIPLE DATASETS

Tutorial for **GeoDict** 2024 SP3

August 23, 2024

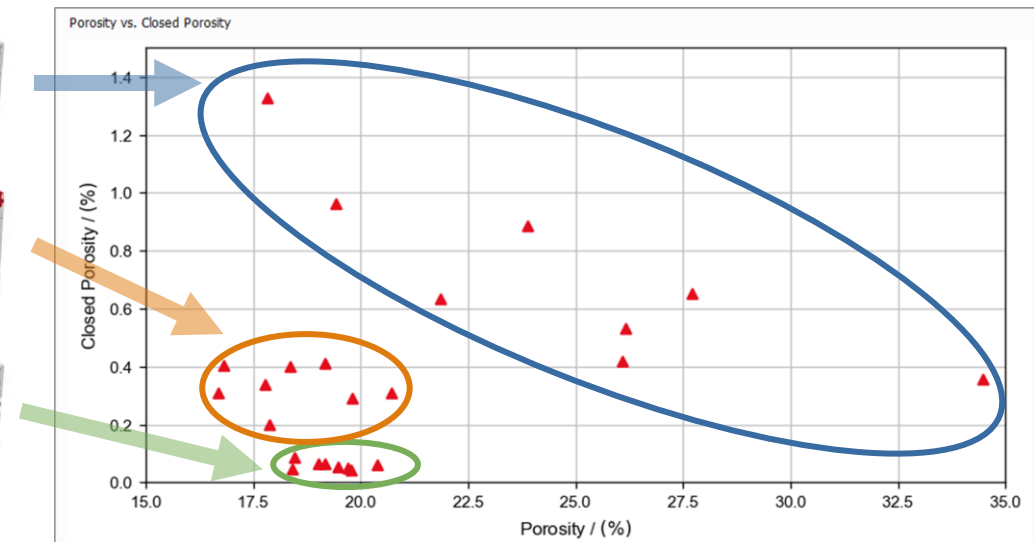
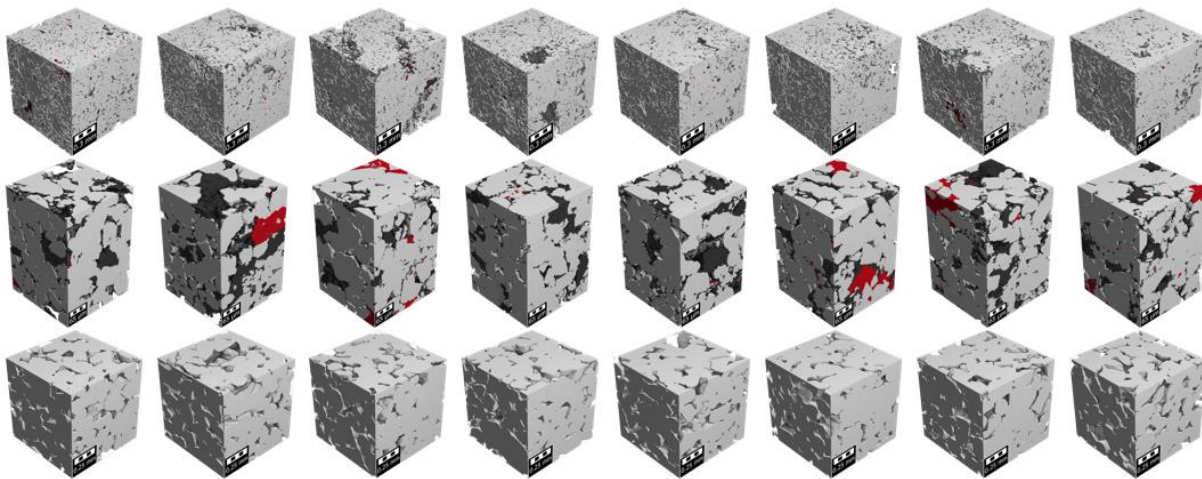
Janine Hilden, Christian Hinz, Anne Blumer, Barbara Planas, Steffen Schwichow

- 1 GeoDict automation possibilities and tutorial set-up
- 2 Record and edit a GeoDict macro to handle multiple datasets
- 3 Summary and outlook

Increase productivity by automating individual workflows.

Automate common and repetitive tasks, perform parameter studies and even the post-processing of the results for a nice presentation. Use the entire range of  Python coding to access any kind of **GeoDict** data.

Find additional help in the workshops available on the **Math2Market's** YouTube channel ([beginners](#) and [advanced](#)) and from the [GeoPy scripting](#) handbook on our website.



What will you learn in this tutorial?

- How to automatically run the same simulation on a batch of structures lying in the same folder.
- How to combine a batch of **GeoDict** result files into one to compare the results manually and automatically.

How to use this tutorial

- The tutorial folder contains three sub-folders:
 - **Input-Data**: contains a batch of **GeoDict** structure files (*.gdt).
 - **Results-M2M**: contains **GeoPy** files (*.py) and results generated by **Math2Market** by following this tutorial.
 - **Results-User**: empty. Use it as project folder and save your results in it.



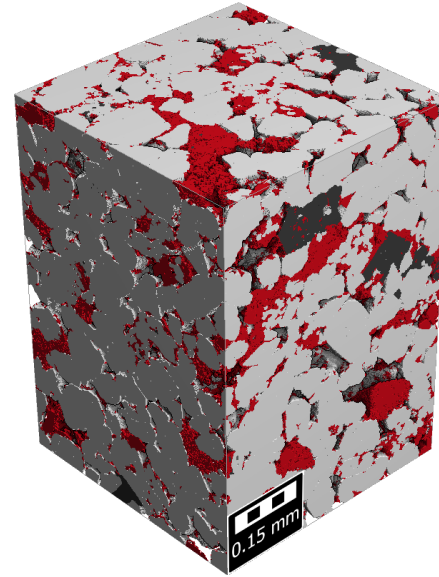
Module needed to follow
this tutorial:
PoroDict

DIGITAL ROCKS USED IN THIS TUTORIAL



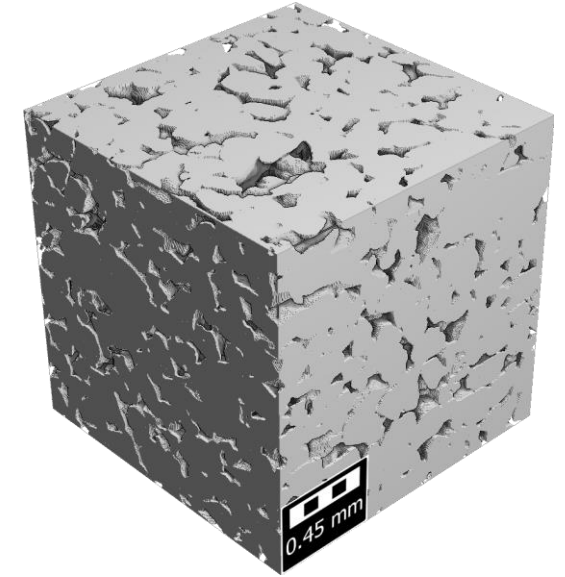
Grossmont carbonate¹

Domain size: 1024x1024x1024 voxels
Voxel length: 2.02 μm



Berea sandstone¹

Domain size: 720x720x1024 voxels
Voxel length: 0.74 μm



Gildehauser sandstone²

Domain size: 400x400x400 voxels
Voxel length: 4.4 μm



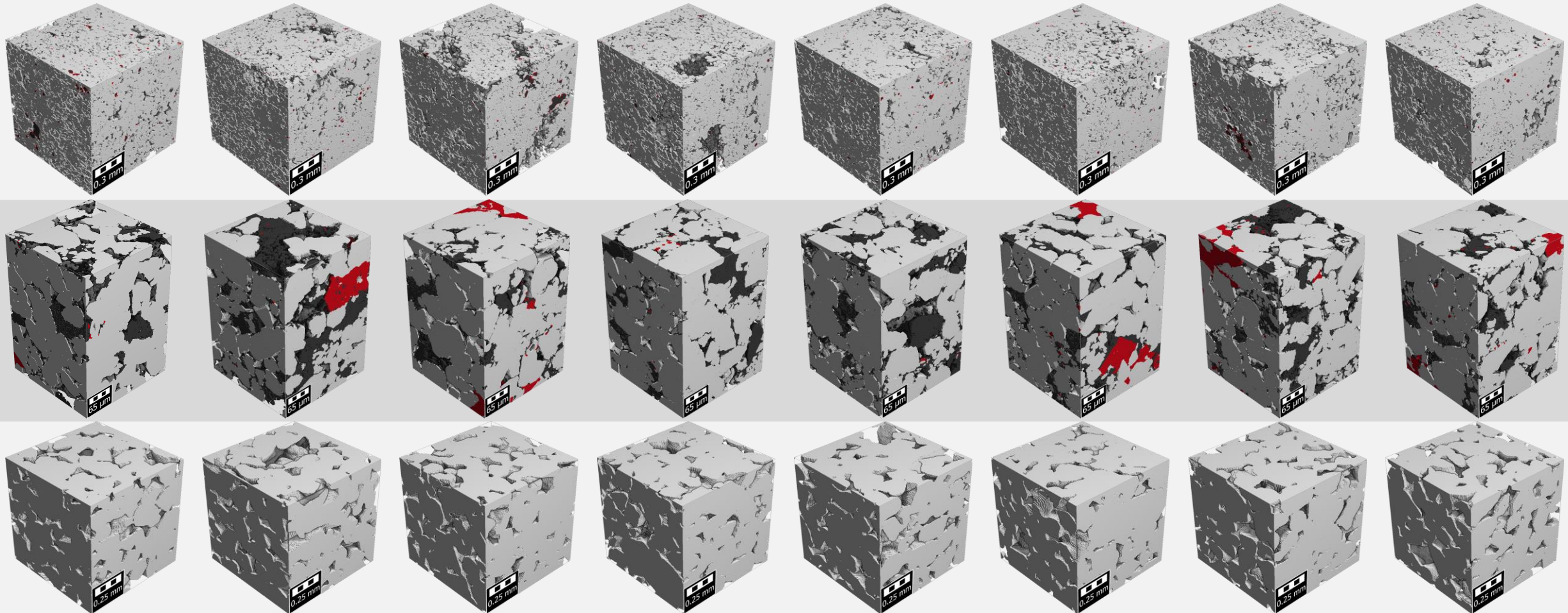
**Segments of three digital rocks are used in the tutorial to illustrate the automation capabilities of GeoDict.
The results are not meant to judge in any way the quality of the data set or physical material.**

¹ Andrä et al., *Digital rock physics benchmarks — Part I: Imaging and segmentation*, (2013) <https://doi.org/10.1016/j.cageo.2012.09.005>

² Berg et al., *Connected pathway relative permeability from pore-scale imaging of imbibition*, (2016) <https://doi.org/10.1016/j.advwatres.2016.01.010>



8 segments are considered for each digital rock.



- Almost every action in **GeoDict** is a command
 - Loading and saving files (structures, results, ...)
 - Starting a simulation
 - Rendering an image/video file
 - ...
- Commands have a name and a set of parameters
 - e.g., result file, boundary condition, solid volume fraction...
- Actions in the graphical user interface (GUI) execute a corresponding command
- A recorded macro is a list of commands with their parameters
- Playing back the macro runs the same commands again
- Macros can have variables, which can be set in the GUI



Watch Math2Market's YouTube channel to learn more about **GeoPy** scripting for beginners and advanced users

or refer to the GeoPy scripting handbook.

- GeoPy is a **Python 3.11** interpreter included in GeoDict.
 - Any regular Python script should run in GeoDict.
 - Can additionally use GeoDict functions (GeoPy API).
 - all start with "gd"
 - Example: gd.runCmd(...)
- **Suggestion:** start off with a recorded Python macro.
- Add variables and Python commands to run customized analyses.
- **Note:** Comments are introduced by #. The rest of the line is ignored.

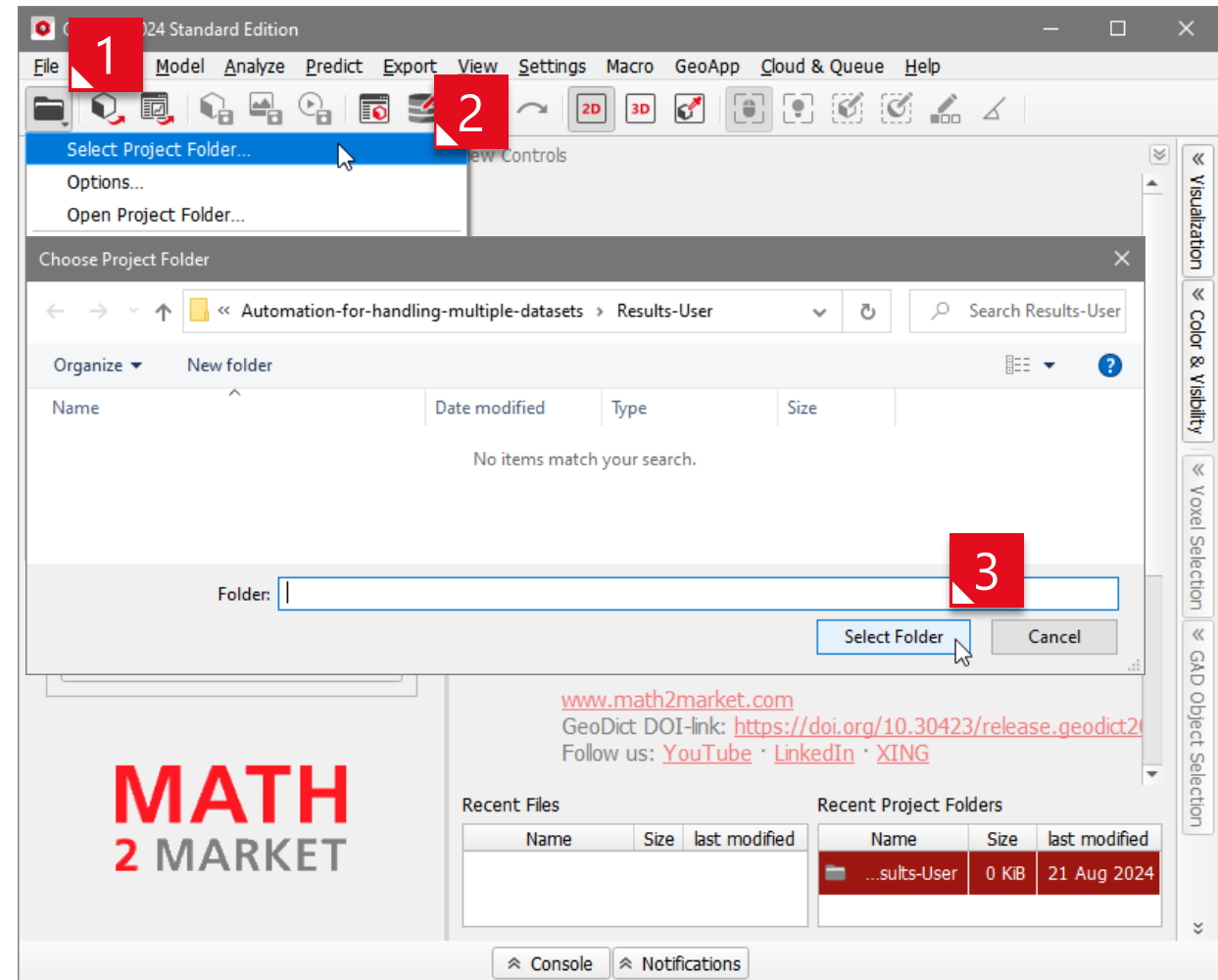
SET UP GEO_DICT PROJECT FOLDER FOR THIS TUTORIAL

Start GeoDict 2024

1. Click the **Choose Project Folder** icon in the toolbar or select **File** → **Choose Project Folder** in the menu bar
2. Click **Select Project Folder...**
3. Navigate to the tutorial folder. Select the **Results-User** folder. Click **Select Folder**



Your files will be saved to this location.
The materials needed for this tutorial are easily loaded from the Tutorial/Input-Data folder.



1 GeoDict automation possibilities and tutorial set-up

2 Record and edit a GeoDict macro to handle multiple datasets

2.1 Part 1 - Create GeoPy script to load structures and compute porosity

2.2 Part 2 - Combine the results and create a custom plot

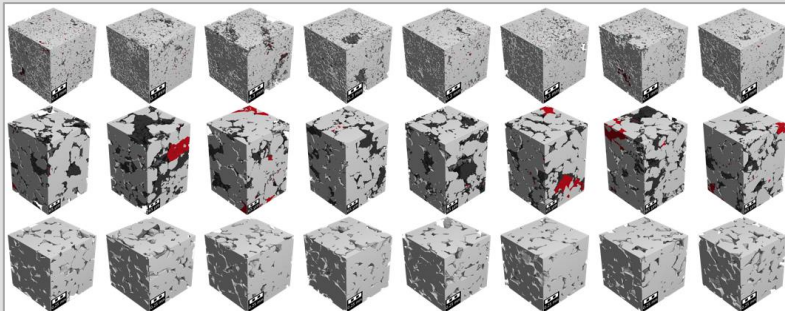
2.3 Part 3 - Automate post-processing and characterize digital rocks

3 Summary and outlook

AUTOMATING THE HANDLING OF MULTIPLE DATASETS

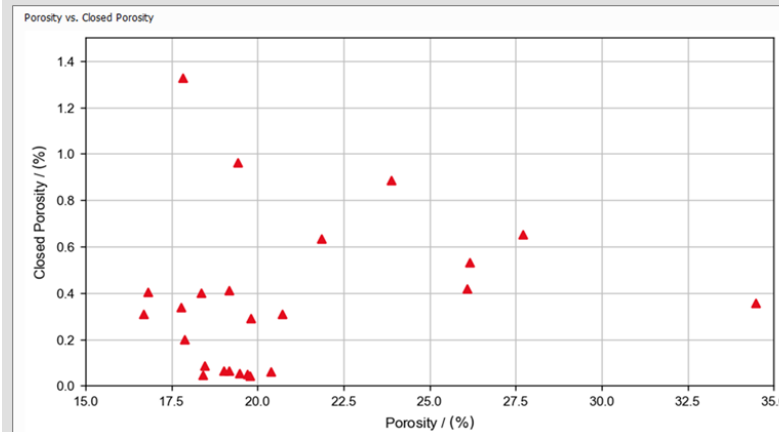
At the end of the tutorial, you will be able to handle multiple datasets automatically and increase productivity when working with GeoDict.

CREATE A **GEO**PY SCRIPT TO LOAD STRUCTURES AND COMPUTE POROSITY



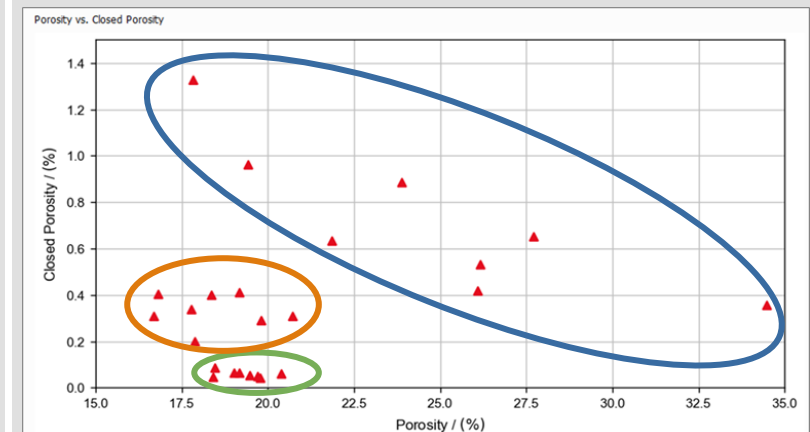
PART 1

COMBINE RESULTS INTO A SINGLE FILE AND CREATE A CUSTOM PLOT



PART 2

AUTOMATE POST-PROCESSING & CHARACTERIZE DIGITAL ROCKS BY POROSITY CLUSTERS



PART 3

1 GeoDict automation possibilities and tutorial set-up

2 Record and edit a GeoDict macro to handle multiple datasets

2.1 Part 1 - Create GeoPy script to load structures and compute porosity

2.2 Part 2 - Combine the results and create a custom plot

2.3 Part 3 - Automate post-processing and characterize digital rocks

3 Summary and outlook

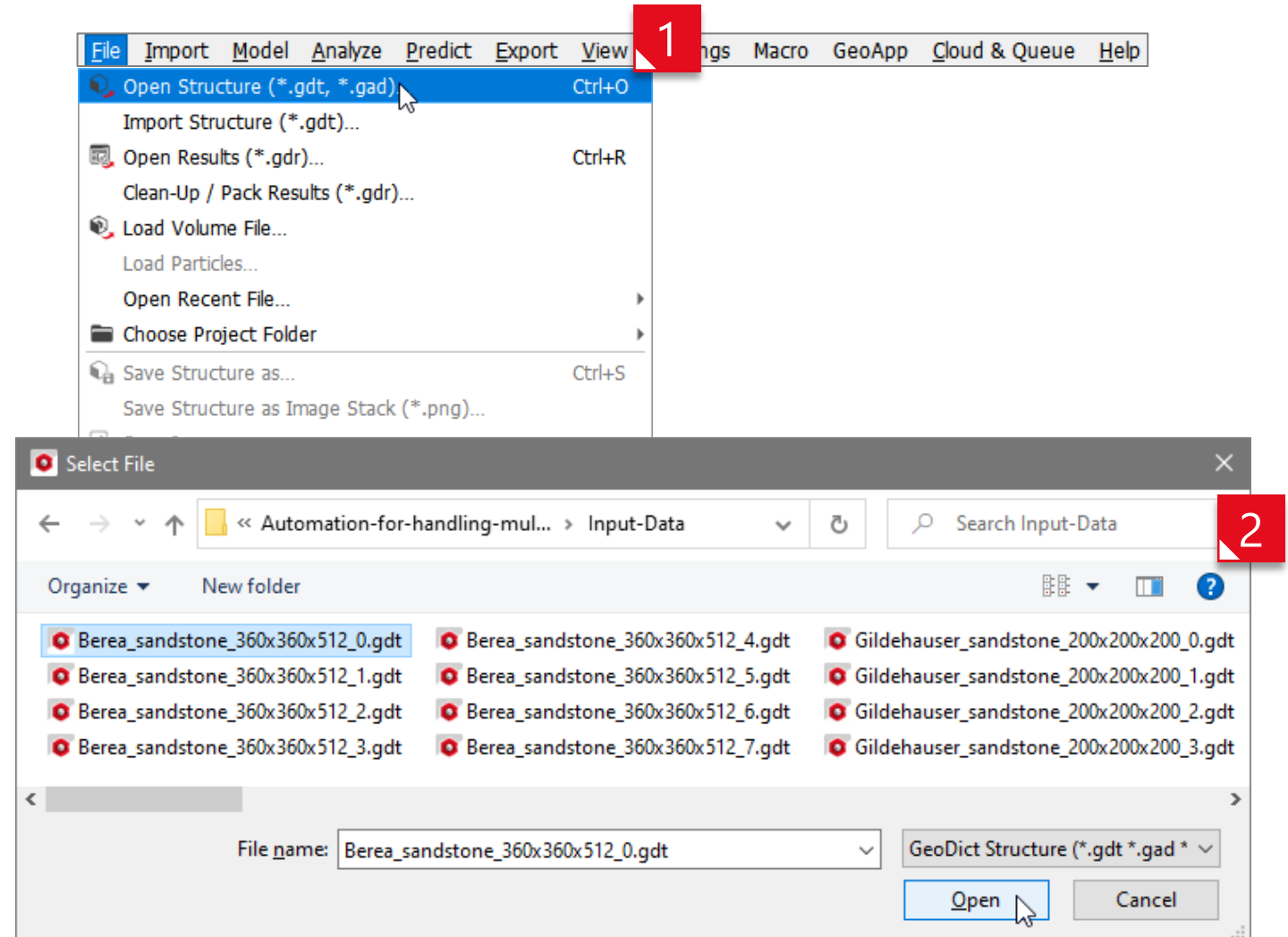
PART 1 – LOAD A STRUCTURE FILE

1. Select **File** → **Open Structure (*.gdt, *.gad)** in the menu bar
2. Navigate to the **Input-Data** folder in the tutorial folder, select any **GeoDict** structure file (*.gdt) and click **Open**

i

gdt: **GeoDict** binary structure file

The sample structures for this tutorial were obtained by importing 3D scans of three different digital rocks into **GeoDict** with the module **ImportGeo-Vol**.



PART 1 – COMPUTE OPEN AND CLOSED POROSITY

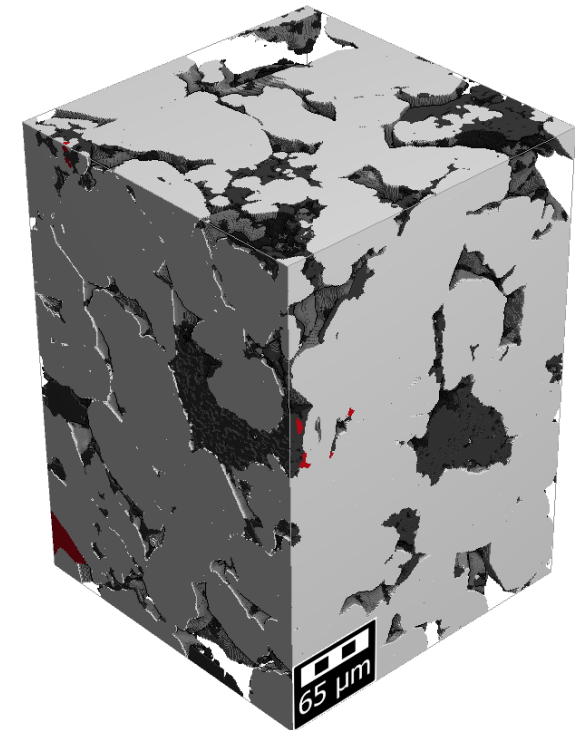
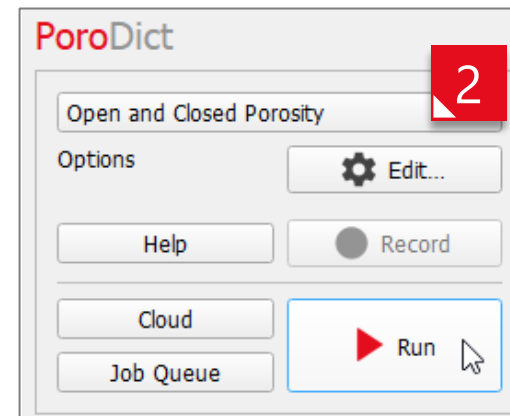
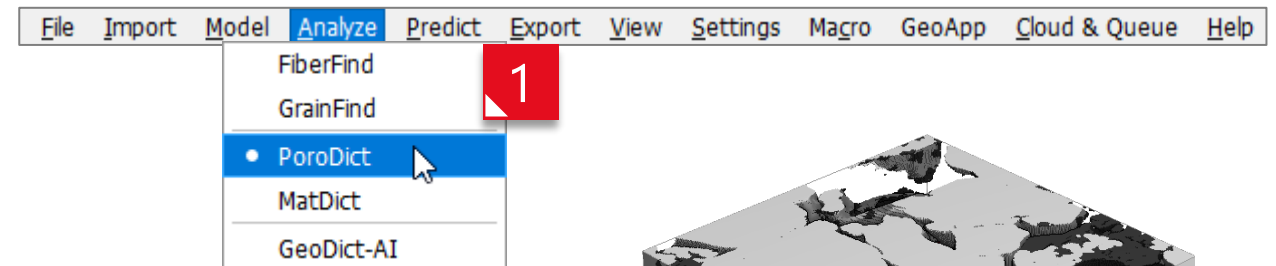
1. Start the **PoroDict** module by selecting **Analyze** → **PoroDict** in the menu bar
2. In the **PoroDict** section, select **Open and Closed Porosity** from the pull-down menu and click **Run**

i

Any other analysis or simulation may be run and recorded to a **GeoPy** macro

i

The color and camera settings for the visualization of the structure may be customized as desired. For more details, see the [Visualization](#) handbook.

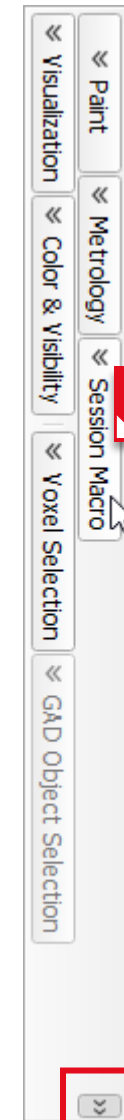
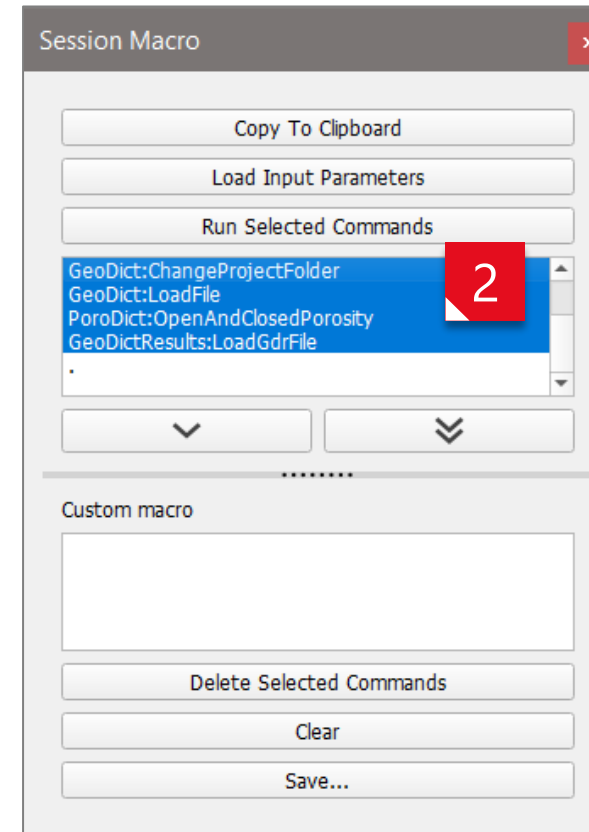


PART 1 – RECORD THE STEPS IN A MACRO FILE

1. Click the **Session Macro** tab in the sidebar, to the right of GUI
 - a) if **GeoDict** is not running in full-screen, the double arrow sign in the bottom right to show all tabs of the side bar
2. Select the needed four commands by holding **Ctrl** and **left-click**
 - a) GeoDict:ChangeProjectFolder
 - b) GeoDict:LoadFile
 - c) PoroDict:OpenAndClosedPorosity
 - d) GeoDictResults:LoadGdrFile



All commands of the running **GeoDict** session are listed in the Session Macro dialog and may be saved to a macro at any time

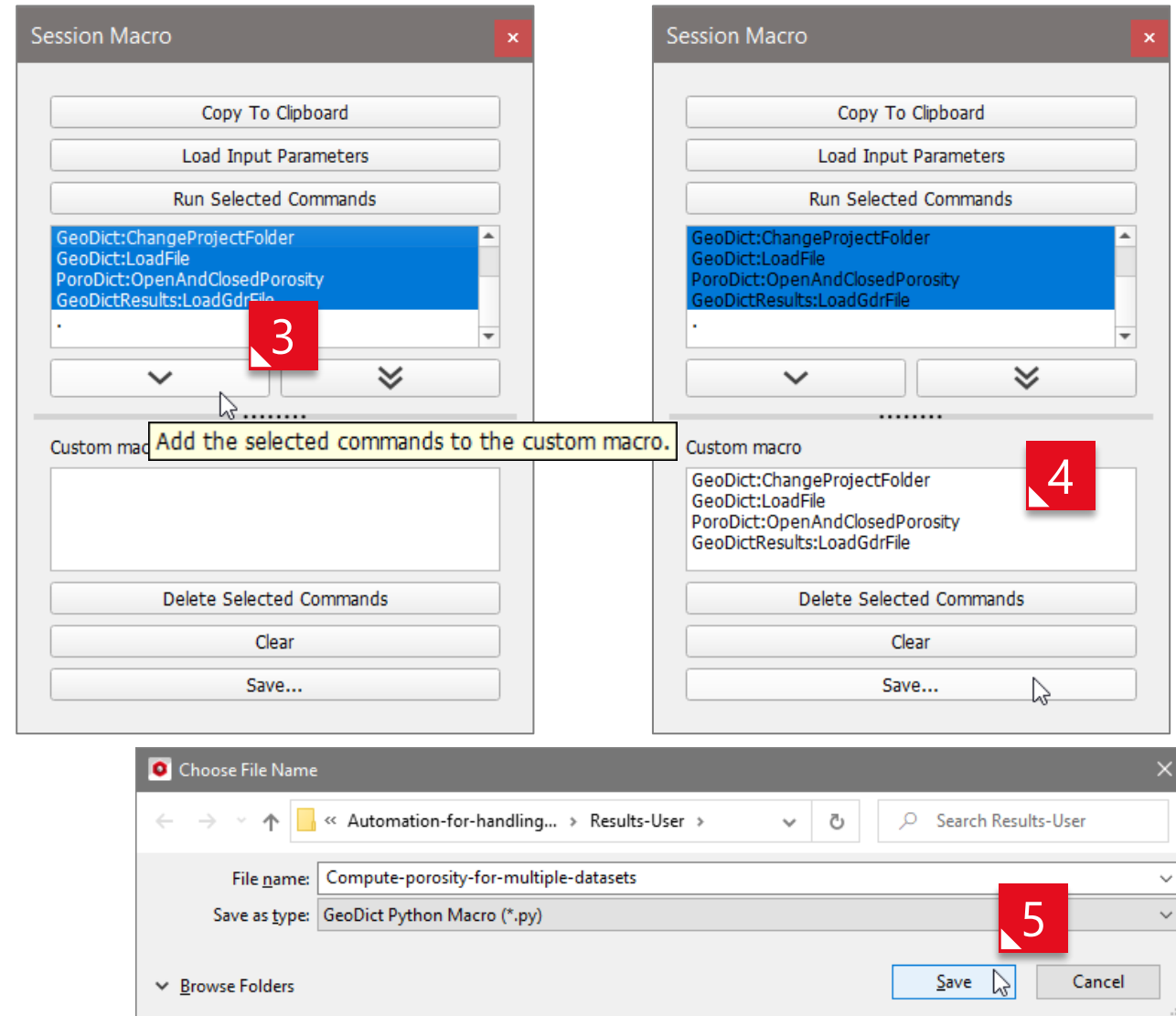


PART 1 – RECORD THE STEPS IN A MACRO FILE

3. Click the single arrow below – the commands appear in the lower panel
4. Click **Save**
5. Enter a name for the macro file, e.g., **Compute-porosity-for-multiple-datasets.py** and click **Save** again



All commands of the running **GeoDict** session are listed in the Session Macro dialog and may be saved to a macro at any time



PART 1 – OPEN AND EDIT THE .PY FILE

1. Open the Macro Execution Control by selecting **Macro** → **Execute Macro / Script** from the menu bar
2. Select the .py file and click **Edit...**

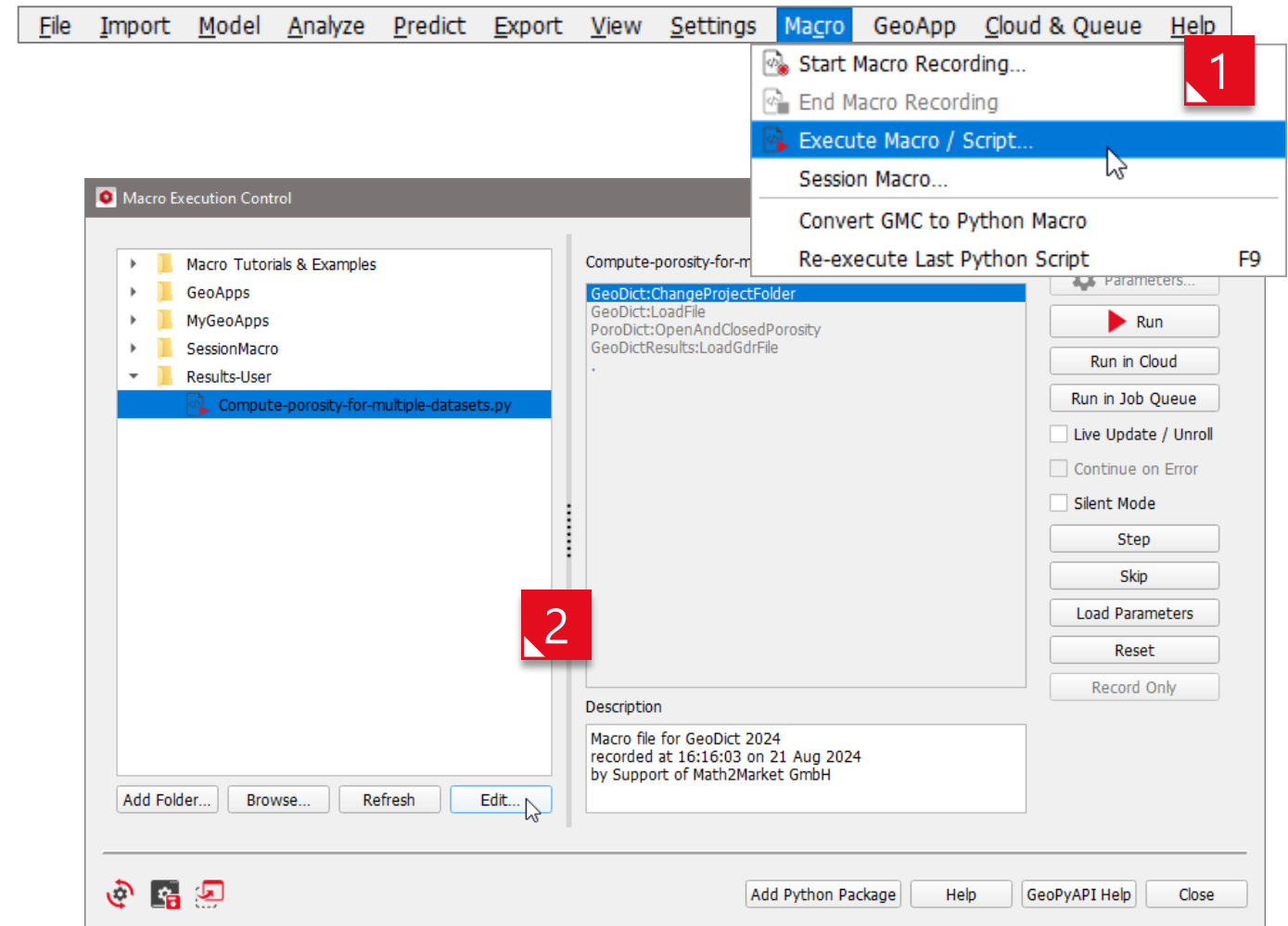
The macro opens in the defined text editor, which by default is Notepad++



To use another text editor, select Settings → Settings in the menu bar and browse for the corresponding executable



The Help button opens the GeoPy scripting handbook. The **GeoPy** API Help button opens an overview of helpful **GeoDict** specific Python functions: to get the domain dimensions, volume field parameters, etc.



In the first line of the .py file, above the header:

1. Type **from glob import glob**
 - a) This module allows to browse for all files matching a given pattern with the given folder, e.g., having the file extension gdt.
2. Below, type **import os**
 - a) This module allows many file and file path manipulations, e.g., to get the given folder name without the path

i

Powerful Python modules are shipped with **GeoDict**: numpy, matplotlib, xlswriter, ...
Additional modules might be installed as described in the [GeoPy scripting handbook](#).

```
1  from glob import glob
2  import os
3
4  Header = {
5      'Release'      : '2024',
6      'Revision'     : '75215',
7      'BuildDate'    : '11 Jun 2024',
8      'CreationDate' : '21 Aug 2024',
9      'CreationTime' : '16:16:03',
10     'Creator'      : 'hilden',
11     'Platform'     : '64 bit Windows',
12 }
```

In the .py file Description section:

- Enter a meaningful text to describe the macro functionality, later displayed in the Macro Execution Control
 - *This macro computes open and closed porosity for all structures in the given folder.*

```
14 Description = '''  
15 This macro computes open and closed porosity for all structures in the given folder.  
16 '''
```

Description

This macro computes open and closed porosity for all structures in the given folder.



Make sure NOT to delete the beginning and ending markers of sections.

The description starts and ends with triple quotes

'''...'''

Python dictionaries start and end with curly brackets - for example, the Variables section

{...}

strings start and end with single or double quotes

'...' or "..."

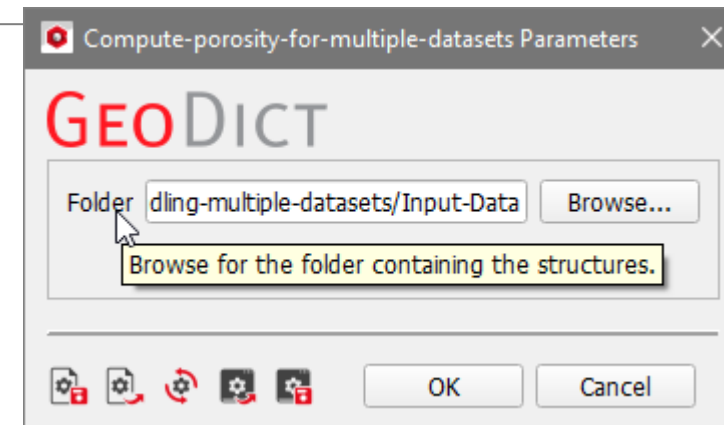
PART 1 – ADD VARIABLE FOR FOLDER PATH

In the Variables section (which controls the parameters accessible from the GUI):

1. Change **NumberOfVariables** from 0 to 1
2. Uncomment the green lines by removing the # symbols
3. Change the name to **gd_folder** and the label to **Folder**
4. The type must be **folderstring**. Then, we get a Browse button in the GUI to find the desired folder path containing the structures
5. The other three parameters are optional and may be omitted. We entered a Tooltip as shown on the right

```
18 Variables = {
19   'NumberOfVariables' : 1,
20   # 'Variable1' : {
21   #   'Name'           : 'gd_SVP',
22   #   'Label'          : 'Solid Volume Percentage',
23   #   'Type'           : 'double',
24   #   'Unit'           : '%',
25   #   'ToolTip'        : 'Solid volume percentage of the created structure.',
26   #   'BuiltinDefault' : 10.0,
27   #   'Check'          : 'min0;max100'
28   # },
29 }
```

```
18 Variables = {
19   'NumberOfVariables' : 1,
20   'Variable1' : {
21     'Name'           : 'gd_folder',
22     'Label'          : 'Folder',
23     'Type'           : 'folderstring',
24     'ToolTip'        : 'Browse for the folder containing the structures.',
25   },
26 }
```



The gd prefix is only a **Math2Market** convention to know that this variable can be set from **GeoDict**. It is NOT necessary for the script to work. The variable name can be chosen freely.

PART 1 – CREATE RESULT FOLDER

Below the variable syntax explanations:

1. Above the **ChangeProjectFolder** parameters, add the following line to get the basename of the folder

```
folder_basename =  
os.path.basename(gd_folder)
```

2. Change the value for **FolderName** in the parameters, as shown in the figure.

Thus, the name of the folder containing the structure files is part of the result folder – here leads to the name *Input-Data_results*

3. Set **CreateIfNotPresent** to **True** to create a new folder



Make sure NOT to delete the commas separating the different entries in the Python dictionary at the end of each line

```
61 | folder_basename = os.path.basename(gd_folder)
62 | ChangeProjectFolder_args_1 = {
63 |     'FolderName'      : folder_basename + '_results',
64 |     'CreateIfNotPresent' : True,
65 | }
66 | gd.runCmd("GeoDict:ChangeProjectFolder", ChangeProjectFolder_args_1, Header['Release'])
```

Below the `ChangeProjectFolder` command:

1. **Indent** the simulation command parameters.
In Notepad++, simply select the sections to be indented and press **tab** on the keyboard.
2. Above write
def run_simulations(filename):

The indented section can be now called by the **run_simulation** command, with the structure file name as input parameter.

Our function contains loading a file, computing open and closed porosity, and loading the resulting GeoDict result file (*.gdr).

i

All commands inside a defined section in Python must have the same indentation level.
The section (here, the function body) ends with the last indented line.

```
68 def run_simulations(filename):
69     LoadFile_args_1 = {
72         gd.runCmd("GeoDict:LoadFile", LoadFile_args_1, Header['Release'])
73     }
74     OpenAndClosedPorosity_args_1 = {
93         gd.runCmd("PorDict:OpenAndClosedPorosity", OpenAndClosedPorosity_args_1, Header['Release'])
94     }
95     LoadGdrFile_args_1 = {
98         gd.runCmd("GeoDictResults:LoadGdrFile", LoadGdrFile_args_1, Header['Release'])

```

In the function body:

1. In the **LoadFile** parameters, change the FileName to **filename**
2. For the simulations, first get the basename of the file without extension by using the module **os** again
file_basename = os.path.basename(filename)
file_basename_withoutextension = os.path.splitext(file_basename)[0]
3. Add the file basename to the resultfilename using +
resultfilename =
file_basename_withoutextension +
'_OpenAndClosedPorosity.gdr'
4. In the **OpenAndClosedPorosity** and the **LoadGdrFile** parameters, change the ResultFileName to **resultfilename**

```
68 def run_simulations(filename):  
69     LoadFile_args_1 = {  
70         'FileName' : filename,  
71     }  
72     gd.runCmd("GeoDict:LoadFile", LoadFile_args_1, Header['Release'])  
73     file_basename = os.path.basename(filename)  
74     file_basename_withoutextension = os.path.splitext(file_basename)[0]  
75     resultfilename = file_basename_withoutextension + '_OpenAndClosedPorosity.gdr'  
76  
77     OpenAndClosedPorosity_args_1 = {  
78         'ResultFileName' : resultfilename,  
79     }  
98     LoadGdrFile_args_1 = {  
99         'ResultFileName' : resultfilename,  
100     }  
101     gd.runCmd("GeoDictResults:LoadGdrFile", LoadGdrFile_args_1, Header['Release'])
```



os.path is used to manipulate file paths.

The function **basename** gets the filename without the folder path.

The function **splitext** splits the file extension from the rest of the file path. Here, this results to ['name', 'gdt'].
By adding [0], we refer to the first entry of the list, which is only the filename without the extension.

In programming, counting usually starts with 0.

In the last line,

1. Add (not indented)

for file in glob(gd_folder + '/*.gdt'):

This starts a loop over all gdt files in the given folder

The star * can be used anywhere in a filepath and is a placeholder for the varying part of the file batch to analyze

2. In the next line, add (indented)

run_simulations(file)

This runs the simulations defined in the function above for each file defined in the loop

1. Click **Save** in NotePad++ to save the changes to the macro file

```
101 | gd.runCmd("GeoDictResults:LoadGdrFile", SdrFile_args_1, Header['Release'])
102 |
103 | for file in glob(gd_folder + '/*.gdt'):
104 |     run_simulations(file)
```



PART 1 – RUN THE MACRO

In the **GeoDict** GUI,

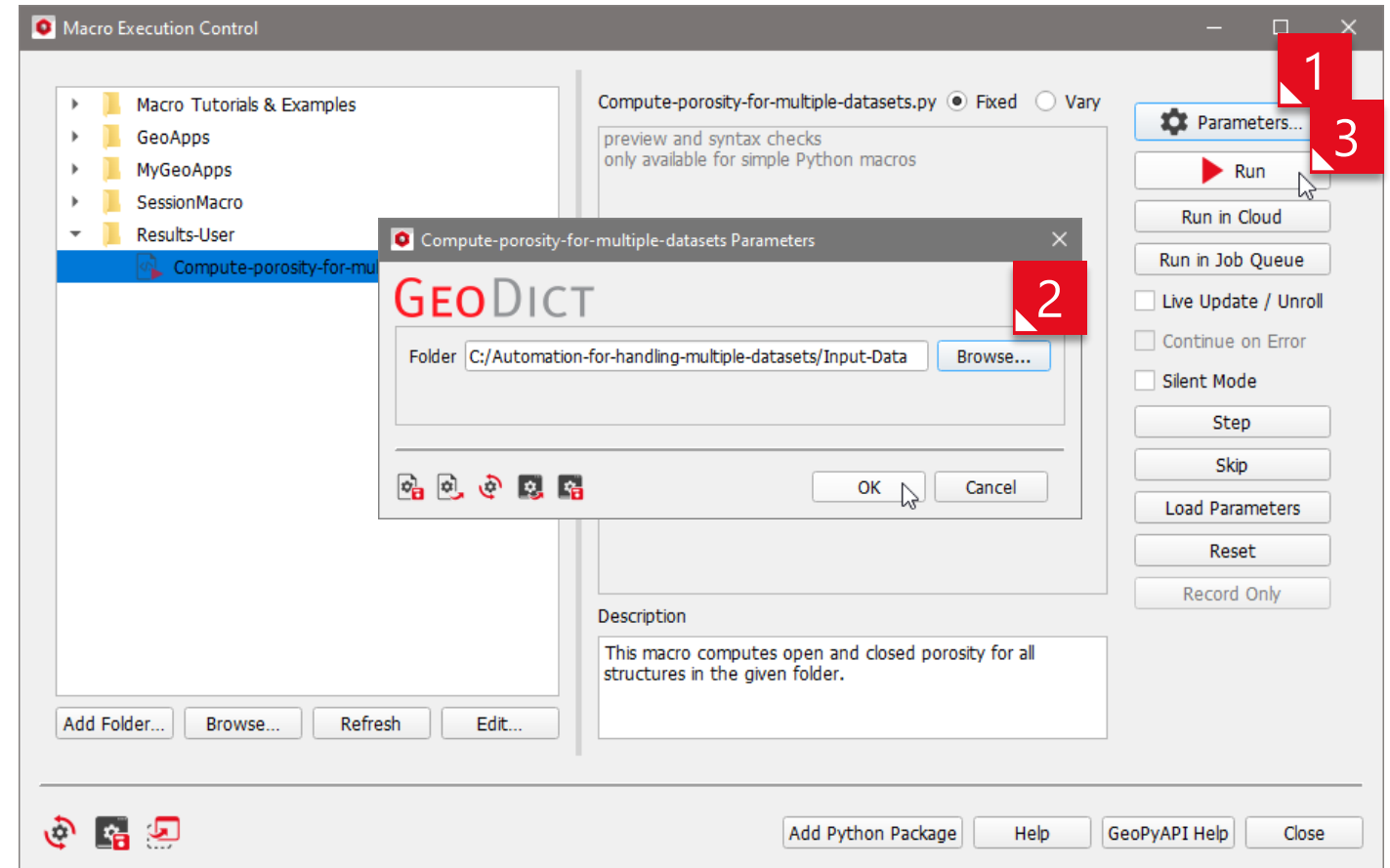
Go back to the Macro Execution Control

- a) **Macro** → **Execute Macro / Script**
- b) or, if it was still open, click **Refresh**

1. Select the macro and click **Parameters**
2. Browse for the **Input-Data** folder inside the tutorial folder and click **OK**
3. Click **Run** to execute the macro

Find the result folder generated by the macro inside the project folder **Results-User**

The result files generated by the macro are opened in the Result Viewer automatically





We have analyzed the open and closed porosity for 24 structure files automatically

Open...

Up Down

Swap Selected

☐ Synchronize Tabs

File	Module	Command
...ea_sandstone_360x360x512_0_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...ea_sandstone_360x360x512_1_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...ea_sandstone_360x360x512_2_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...ea_sandstone_360x360x512_3_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...ea_sandstone_360x360x512_4_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...ea_sandstone_360x360x512_5_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...ea_sandstone_360x360x512_6_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...ea_sandstone_360x360x512_7_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...er_sandstone_200x200x200_0_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...er_sandstone_200x200x200_1_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...er_sandstone_200x200x200_2_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...er_sandstone_200x200x200_3_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...er_sandstone_200x200x200_4_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...er_sandstone_200x200x200_5_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...er_sandstone_200x200x200_6_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...er_sandstone_200x200x200_7_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...nt_carbonate_512x512x512_0_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...nt_carbonate_512x512x512_1_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...nt_carbonate_512x512x512_2_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...nt_carbonate_512x512x512_3_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...nt_carbonate_512x512x512_4_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...nt_carbonate_512x512x512_5_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...nt_carbonate_512x512x512_6_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity
...nt_carbonate_512x512x512_7_OpenAndClosedPorosity.gdr	PoroDict	OpenAndClosedPorosity

GEODict

Combine Results

Raise Main Window

MI Aug 21 2024 (2024 Build 75215)

.../Input-Data_results/Berea_sandstone_360x360x512_0_OpenAndClosedPorosity.gdr

Domain Size: 360 x 360 x 512 Voxel Length: 740 nm

Load Structure

Input Map Log Map Results Data Visualization Metadata

Report Map

Open and closed porosities

Overall porosity / (%)	17.8884
Open porosity / (%)	17.6887
Closed porosity / (%)	0.199642
Number of pores	10782
Number of open pores	94
Number of closed pores	10688

Dead-end and through pores

The open pores can be sub-classified in:

- dead-end pores, which have only contact with one of the specified inflow planes.
- through pores, which connect multiple specified inflow planes.

Manage Data

Load Input Parameters

Export

Close

1 GeoDict automation possibilities and tutorial set-up

2 Record and edit a GeoDict macro to handle multiple datasets

2.1 Part 1 - Create GeoPy script to load structures and compute porosity

2.2 Part 2 - Combine the results and create a custom plot

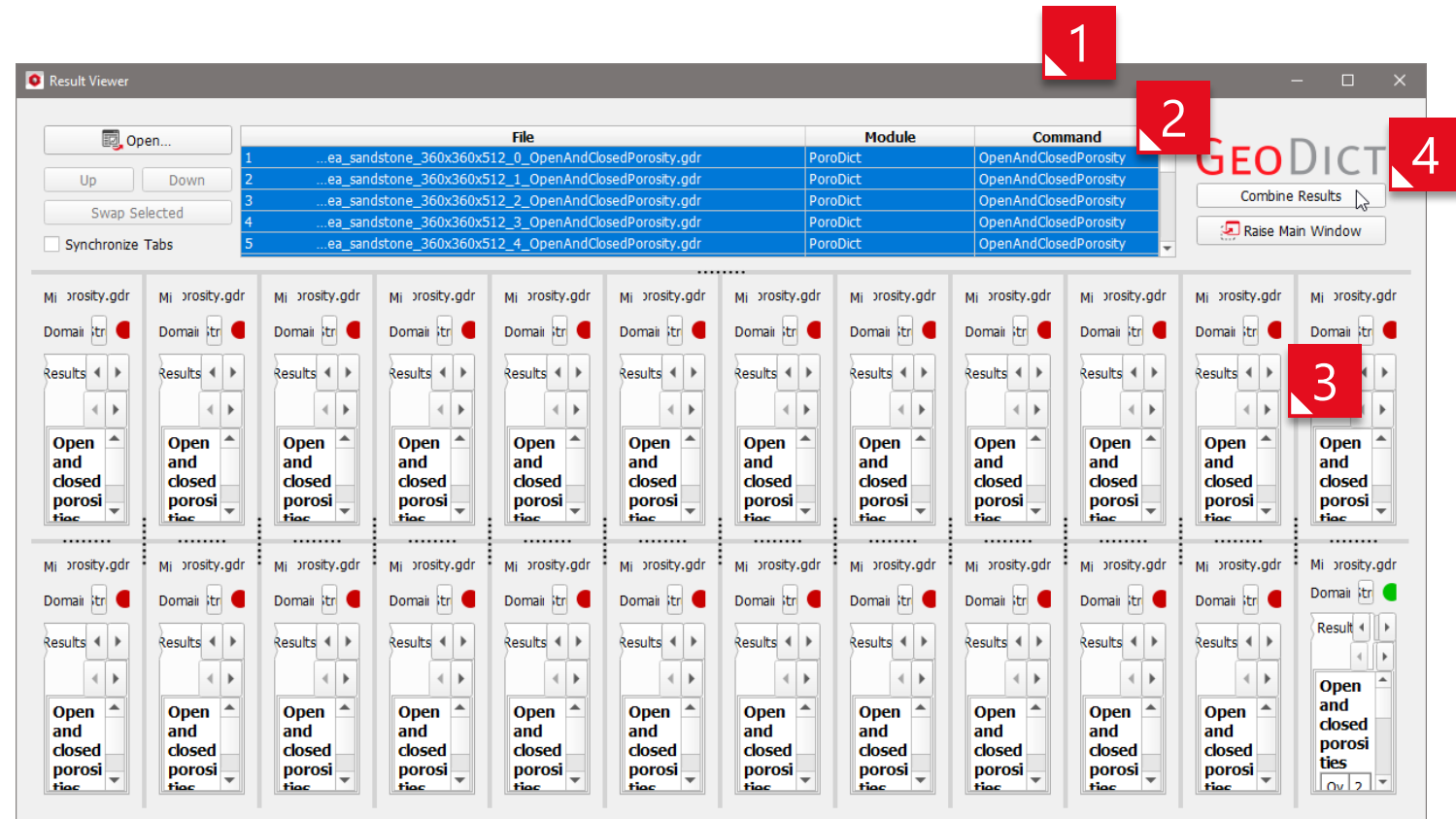
2.3 Part 3 - Automate post-processing and characterize digital rocks

3 Summary and outlook

PART 2 – COMBINE THE RESULTS

In the Result Viewer,

1. In the **header box** at the top, select the first result file
Berea_sandstone_360x360x512_0_OpenAndClosedPorosity.gdr
2. Scroll down, press and hold **Shift**, and select the last result file
Grossmont_carbonate_512x512x512_7_OpenAndClosedPorosity.gdr
3. Now, all 24 result files are selected and displayed
4. To get all results into one file, click **Combine Results** at the top right



PART 2 – COMBINE THE RESULTS

In the Combine Results dialog,

1. Enter **Compare-porosity-for-rocks.gdr** as Result File Name
2. Click **OK**

All results are now combined into a single result file.

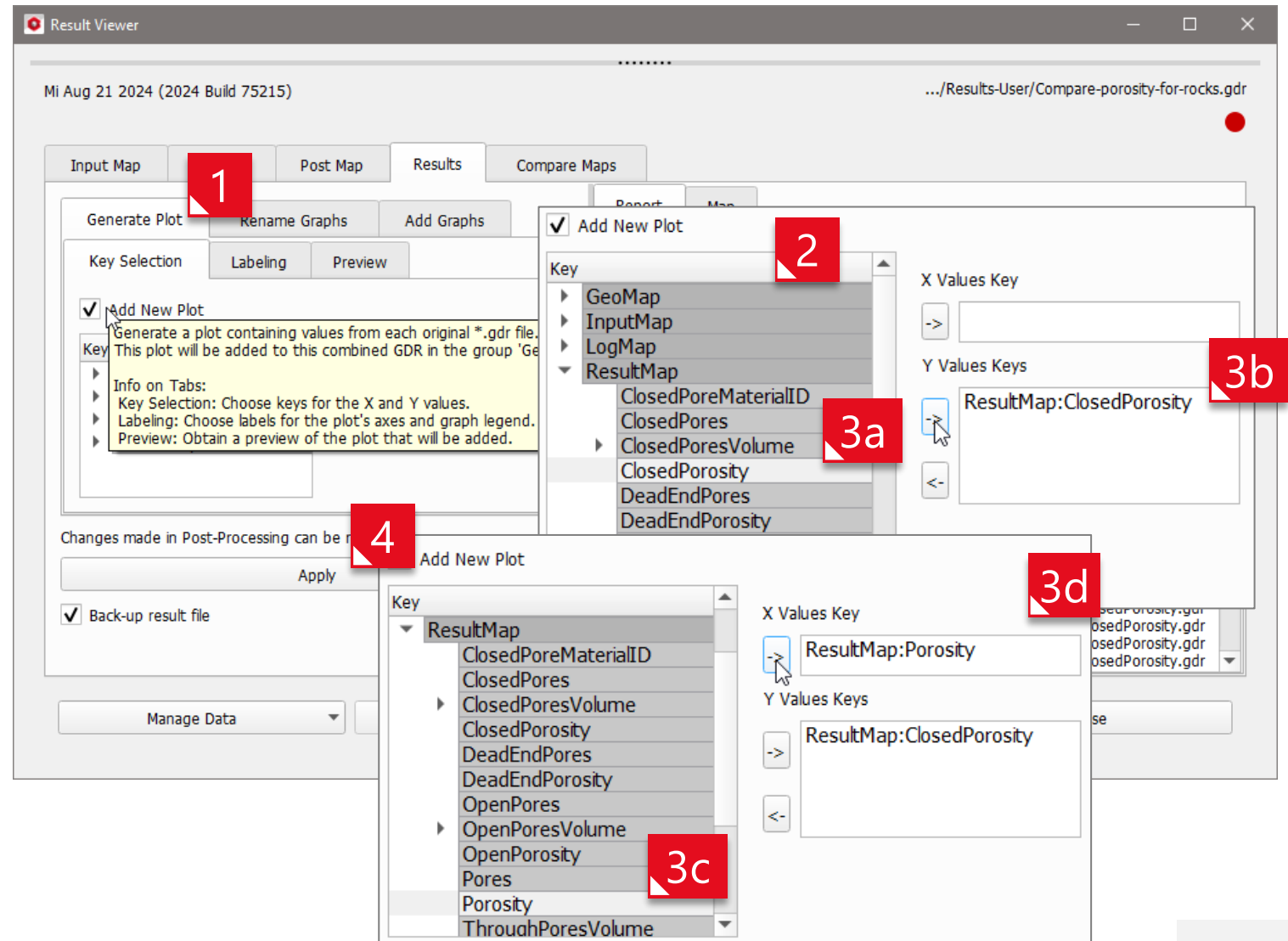
Let's plot closed porosity vs. overall porosity.



PART 2 – ADD CUSTOM PLOT

Under the Results tab,

1. Check **Add New Plot**
2. The plotting keys to be used appear
3. Unfold the **ResultMap** parameters
 - a) Select **ClosedPorosity**
 - b) Click the arrow for **Y Values Keys**
 - c) Select **Porosity**
 - d) Click the arrow for **X Values Keys**
4. Click **Apply**

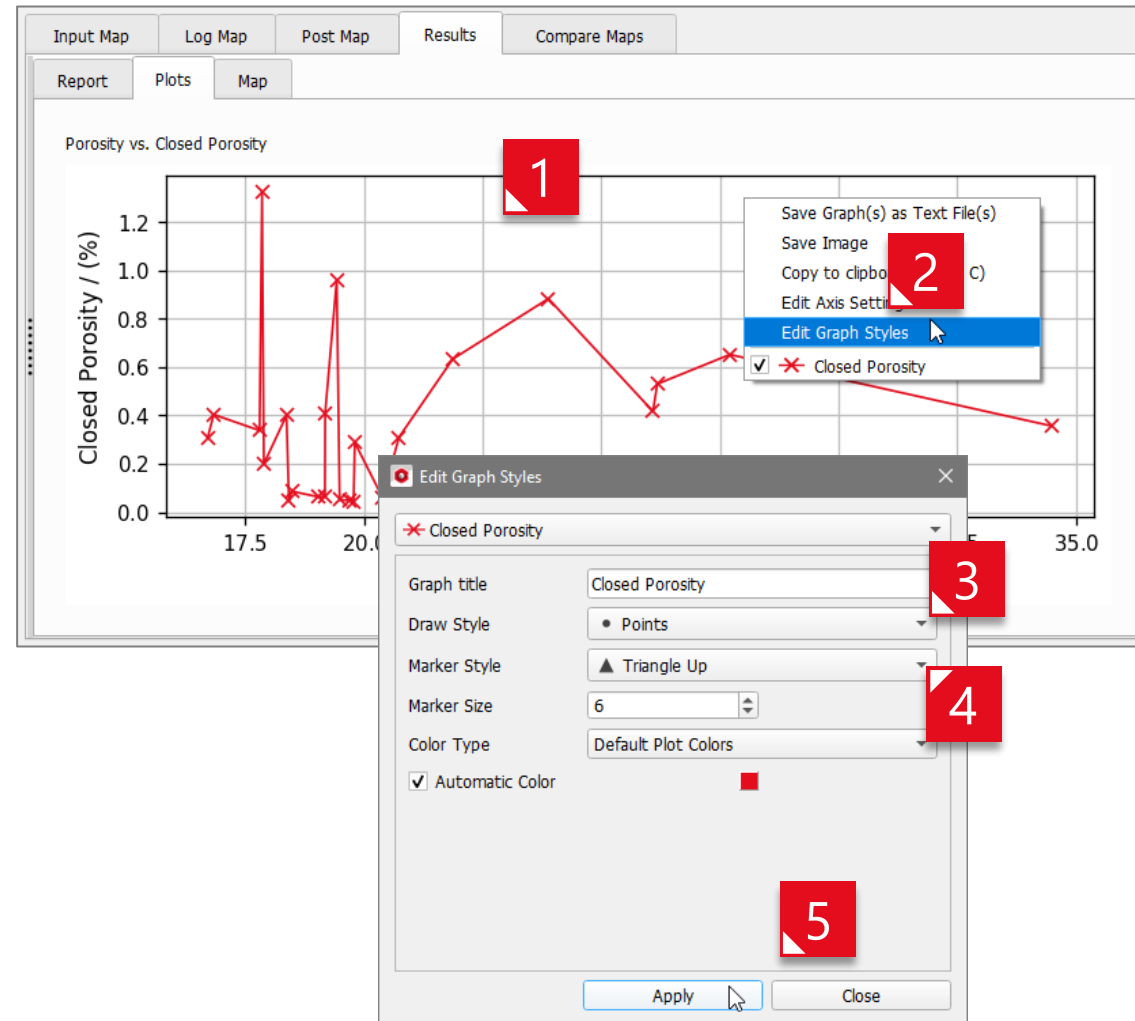


PART 2 – CHANGE GRAPH STYLES

Under the Results - Plots subtab,

- Find the combined plot
- Let's get a scatter plot, instead of connecting the data points with lines

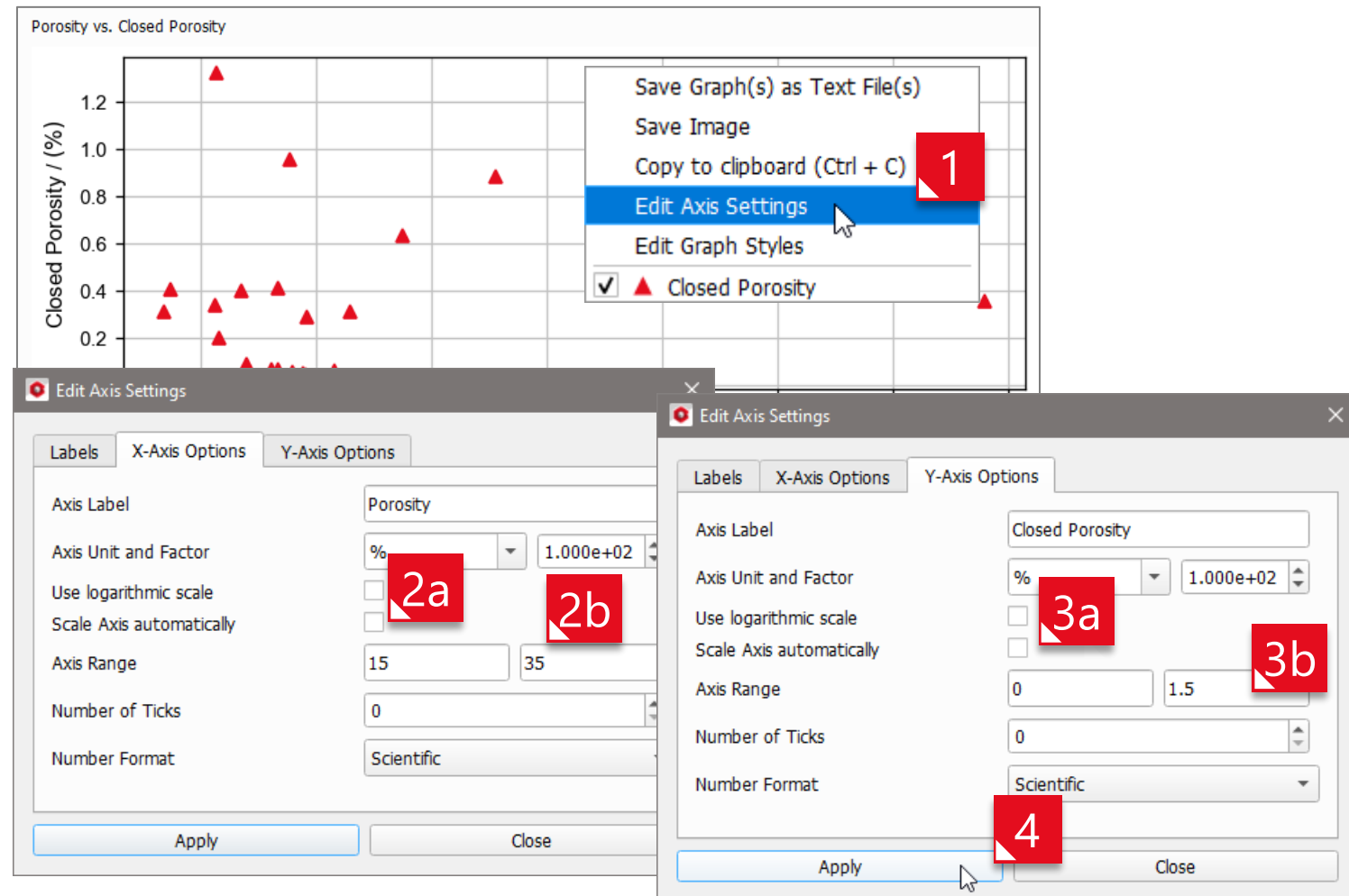
1. Right-click in the plot
2. Select **Edit Graph Styles**
3. Select **Points** as the **Draw Style**
4. Select **Triangle Up** as the **Marker Style**
5. Click **Apply** and then, click **Close**



PART 2 – CHANGE AXIS SETTINGS

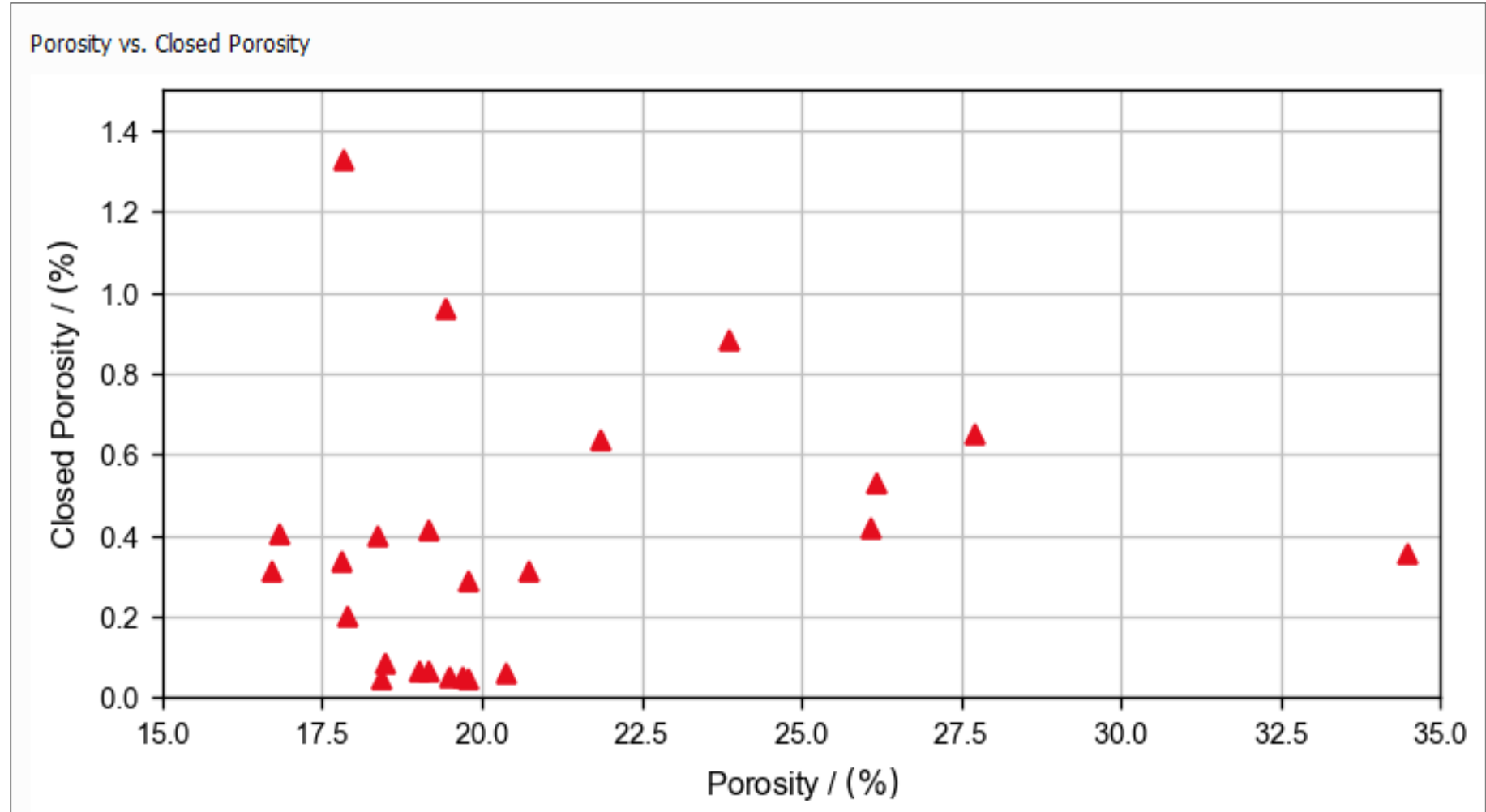
Under the Results - Plots subtab,

1. Again **right-click** in the plot and select **Edit Axis Settings**
2. Under the **X-Axis Options**
 - a) Uncheck **Scale Axis automatically**
 - b) Set the **Axis Range** to **15-35**
3. Under the **Y-Axis Options**
 - a) Uncheck **Scale Axis automatically**
 - b) Set the **Axis Range** to **0-1.5**
4. Click **Apply** and **Close**





The results for 24 digital rock segments are plotted into one graph with customized plot settings



1 GeoDict automation possibilities and tutorial set-up

2 Record and edit a GeoDict macro to handle multiple datasets

2.1 Part 1 - Create GeoPy script to load structures and compute porosity

2.2 Part 2 - Combine the results and create a custom plot

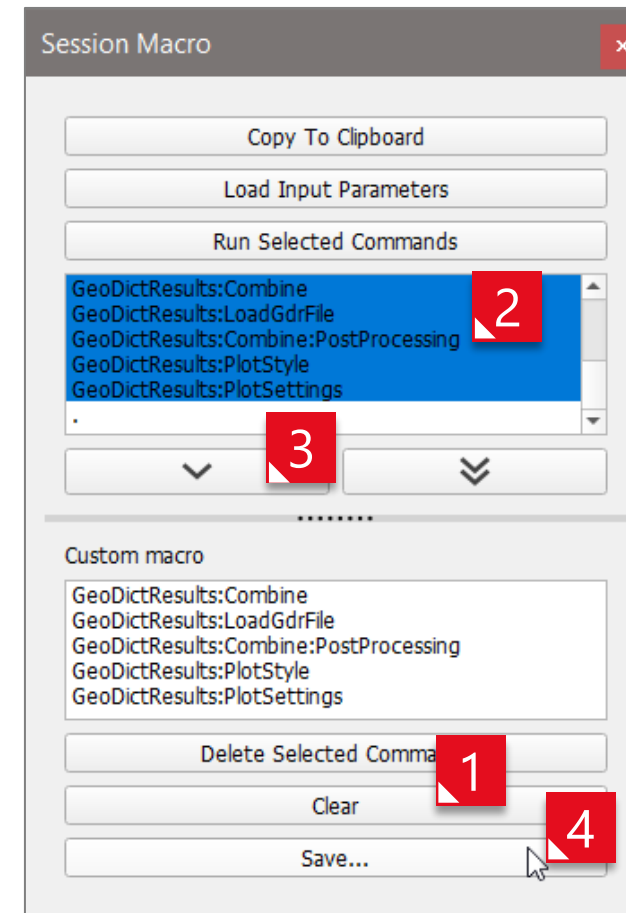
2.3 Part 3 - Automate post-processing and characterize digital rocks

3 Summary and outlook

PART 3 – AUTOMATE COMBINED GDR AND PLOTTING

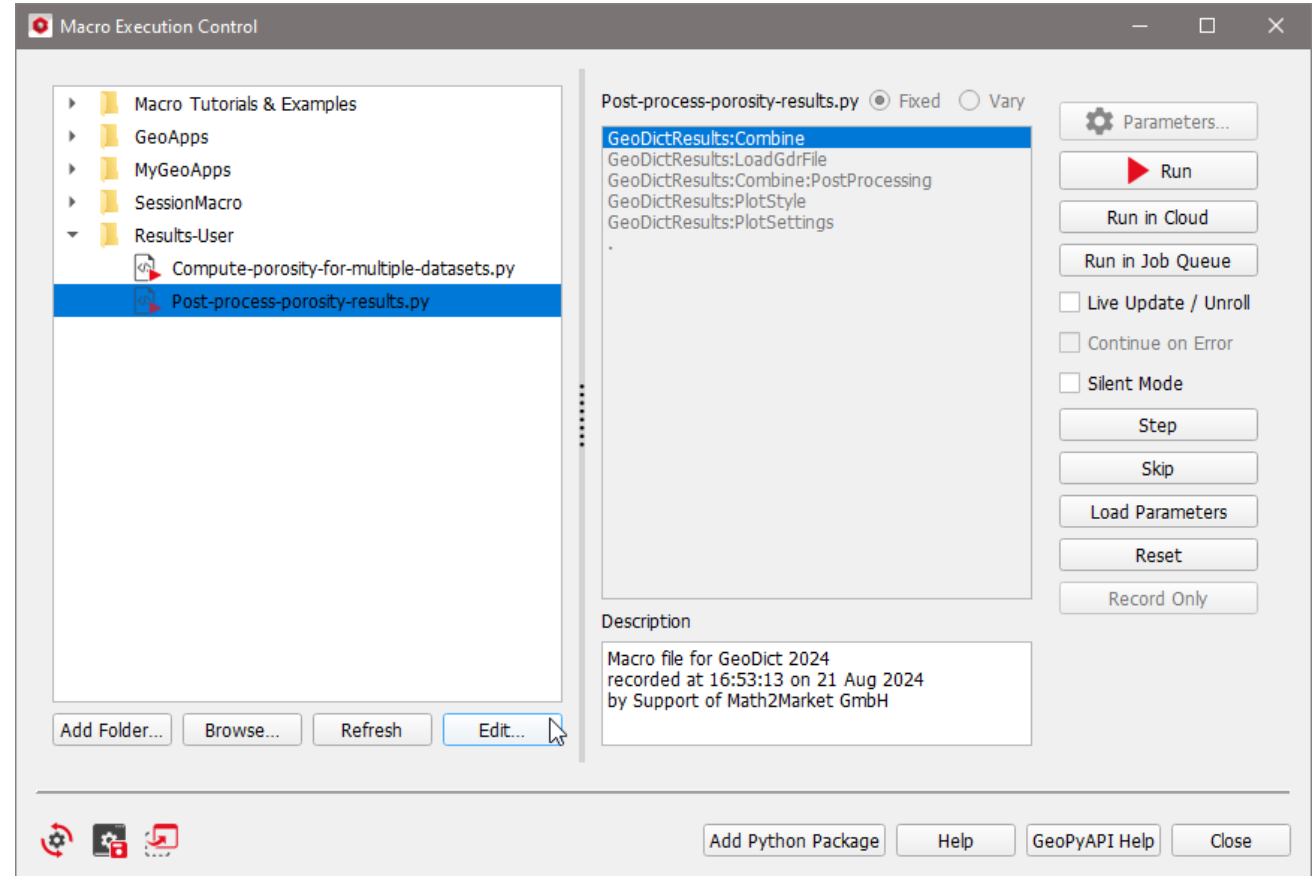
In the **Session Macro** tab of the **GeoDict GUI sidebar**,

1. Click **Clear** to remove the commands saved in the first macro from the lower panel
 - a) Click **Yes** in the opening dialog to confirm clearing the panel
2. Select the commands
 - a) GeoDictResults:Combine
 - b) GeoDictResults:LoadGdrFile
 - c) GeoDictResults:Combine:PostProcessing
 - d) GeoDictResults:PlotStyle
 - e) GeoDictResults:PlotSettings
3. Click the single arrow
4. Click **Save**
 - a) Enter the name **Post-process-porosity-results.py**



PART 3 – OPEN THE MACRO

- Open the Macro Execution Control by selecting **Macro → Execute Macro /Script** from the menu bar and click **Refresh**
- Select the new .py file and click **Edit...**



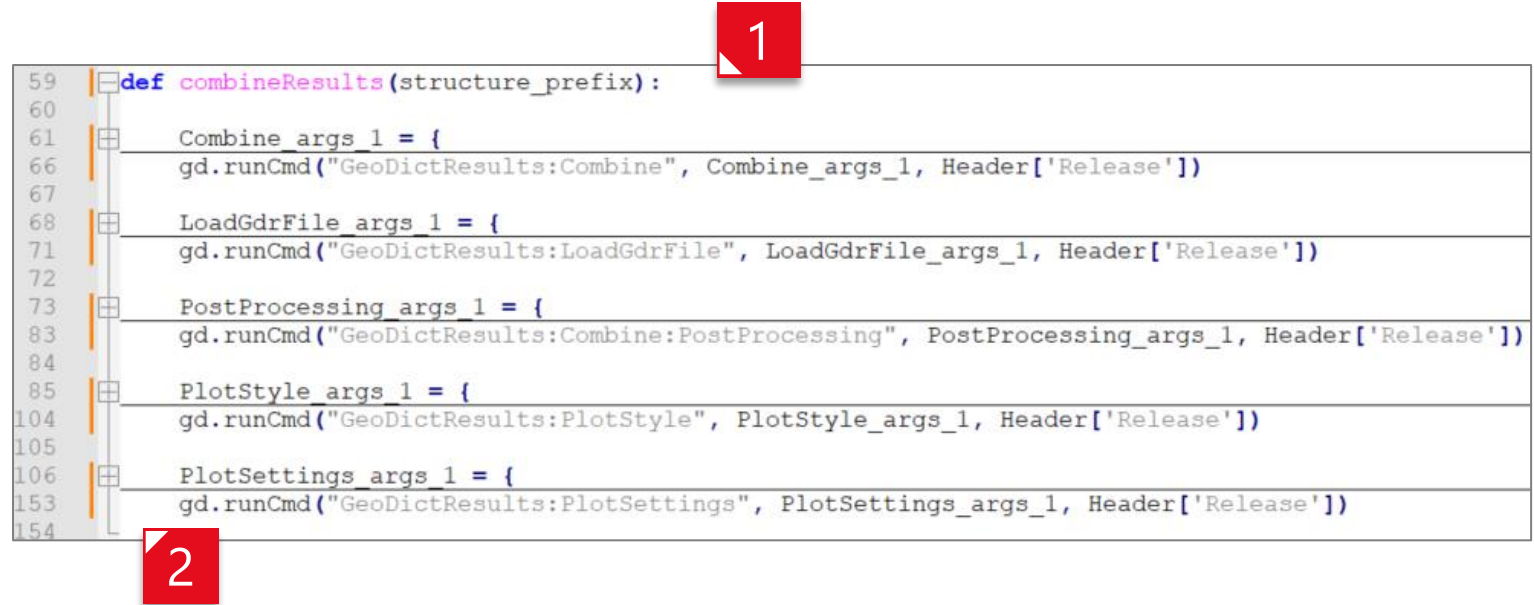
In the text editor,

- Add **from glob import glob** in the first line
- Add a description, e.g.
 - *This macro combines open and closed porosity result files in the given folder.*
- Copy the **Variables** block from the first .py macro and paste it here. Maybe change the tooltip a bit

```
1  from glob import glob
2
3  Header = {
4      'Release'      : '2024',
5      'Revision'     : '75215',
6      'BuildDate'    : '11 Jun 2024',
7      'CreationDate' : '21 Aug 2024',
8      'CreationTime' : '16:53:13',
9      'Creator'      : 'hilden',
10     'Platform'     : '64 bit Windows',
11 }
12
13 Description = '''
14     This macro combines open and closed porosity result files in the given folder.
15 '''
16
17 Variables = {
18     'NumberOfVariables' : 1,
19     'Variable1' : {
20         'Name'          : 'gd_folder',
21         'Label'         : 'Folder',
22         'Type'          : 'folderstring',
23         'ToolTip'       : 'Browse for the folder containing the results.',
24     },
25 }
```

Wrap all commands into a function:

1. Above the commands add
def combineResults(structure_prefix):
2. **Indent** all commands



```
59 def combineResults(structure_prefix):
60
61     Combine_args_1 = {
66         gd.runCmd("GeoDictResults:Combine", Combine_args_1, Header['Release'])
67     }
68
69     LoadGdrFile_args_1 = {
71         gd.runCmd("GeoDictResults:LoadGdrFile", LoadGdrFile_args_1, Header['Release'])
72     }
73
74     PostProcessing_args_1 = {
83         gd.runCmd("GeoDictResults:Combine:PostProcessing", PostProcessing_args_1, Header['Release'])
84     }
85
86     PlotStyle_args_1 = {
104         gd.runCmd("GeoDictResults:PlotStyle", PlotStyle_args_1, Header['Release'])
105     }
106
107     PlotSettings_args_1 = {
153         gd.runCmd("GeoDictResults:PlotSettings", PlotSettings_args_1, Header['Release'])
154     }
```

Define result file name:

1. Above the commands add indented **resultfilename = f'Compare-porosity-for-rocks_{structure_prefix}.gdr'**
2. Change the ResultFileName or name of the PostProcessingFile for all commands by replacing the value with **resultfilename**

```
59 def combineResults(structure_prefix):  
60     resultfilename = f'Compare-porosity-for-rocks_{structure_prefix}.gdr'  
61  
62     Combine_args_1 = {  
63         'ResultFileName' : resultfilename,  
64     }  
65  
66     gd.runCmd("GeoDictResults:CombineResults", Combine_args_1, Header['Release'])  
67  
68  
69     LoadGdrFile_args_1 = {  
70         'ResultFileName' : resultfilename,  
71     }  
72     gd.runCmd("GeoDictResults:LoadGdrFile", LoadGdrFile_args_1, Header['Release'])  
73  
74  
75     PostProcessing_args_1 = {  
76         'PostProcessingFile' : resultfilename,  
77     }  
78     gd.runCmd("GeoDictResults:PostProcessing", PostProcessing_args_1, Header['Release'])  
79  
80  
81  
82  
83  
84  
85  
86     PlotStyle_args_1 = {  
87         'ResultFileName' : resultfilename,  
88     }  
89  
90     gd.runCmd("GeoDictResults:PlotStyle", PlotStyle_args_1, Header['Release'])  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151     'ResultFileName' : resultfilename,  
152     'PlotIdentifier' : 'generated plots:porosity vs. closed porosity',  
153 }  
154 gd.runCmd("GeoDictResults:PlotSettings", PlotSettings_args_1, Header['Release'])
```

Define files to combine:

1. In the parameters for **Combine**, delete the long GDRList and replace it by
glob(gd_folder + '/' + structure_prefix + '*OpenAndClosedPorosity.gdr')
2. Move the **LoadGdrFile** parameters and command at the end of the function. Otherwise, the post-processing is not updated.
Make sure to keep the indentation level.
3. Below the function body, add (not indented) the three function calls for the three digital rocks
combineResults('Berea')
combineResults('Gildehauser')
combineResults('Grossmont')
4. Click **Save** in Notepad++ to save the changes to the macro file.

```
59 def combineResults(structure_prefix):
60     resultfilename = f'Compare-porosity-for-rocks_{structure_prefix}.gdr'
61
62     Combine_args_1 = {
63         'ResultFileName' : resultfilename,
64         'GdrList'       : glob(gd_folder + '/' + structure_prefix + '*OpenAndClosedPorosity.gdr'),
65         'CombineMode'   : 'Color', # Possible values: Color, Marker
66     }
67     gd.runCmd("GeoDictResults:Combine", Combine_args_1, Header['Release'])
68
69     PostProcessing_args_1 = {
70         'PostProcessing' : 'Combine',
71     }
72     gd.runCmd("GeoDictResults:Combine:PostProcessing", PostProcessing_args_1, Header['Release'])
73
74     PlotStyle_args_1 = {
75         'PlotStyle' : 'Combine',
76     }
77     gd.runCmd("GeoDictResults:PlotStyle", PlotStyle_args_1, Header['Release'])
78
79     PlotSettings_args_1 = {
80         'PlotSettings' : 'Combine',
81     }
82     gd.runCmd("GeoDictResults:PlotSettings", PlotSettings_args_1, Header['Release'])
83
84     LoadGdrFile_args_1 = {
85         'LoadGdrFile' : 'Combine',
86     }
87     gd.runCmd("GeoDictResults:LoadGdrFile", LoadGdrFile_args_1, Header['Release'])
88
89     combineResults('Berea')
90     combineResults('Gildehauser')
91     combineResults('Grossmont')
```

PART 3 – RUN THE MACRO

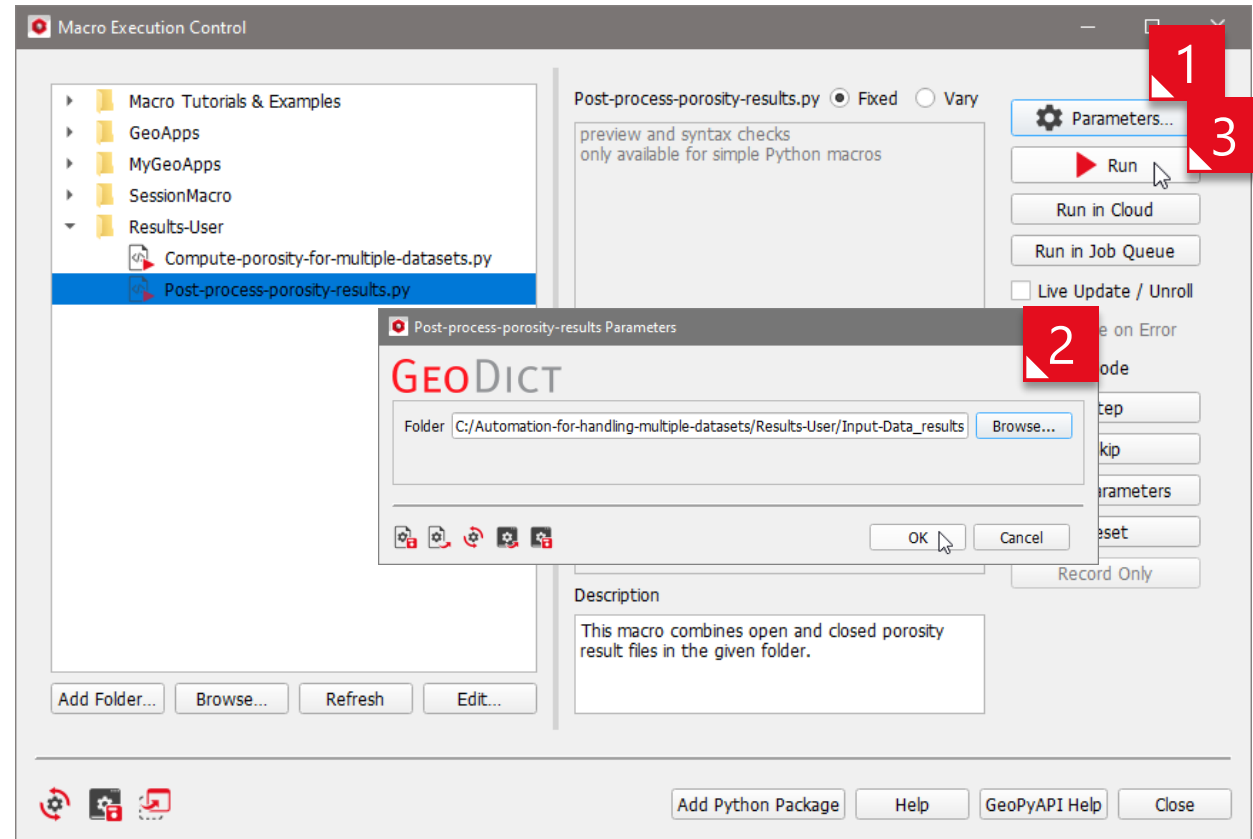
In the **GeoDict GUI**,

Go back to the Macro Execution Control by selecting **Macro → Execute Macro / Script** in the menu bar and click **Refresh**

1. Select the .py macro and click **Parameters**.
2. Browse for the **Input-Data_results** folder containing the result files and click **OK**.
3. Click **Run** to execute the macro.

Find the results generated by the macro inside the project folder (**Results-User**).

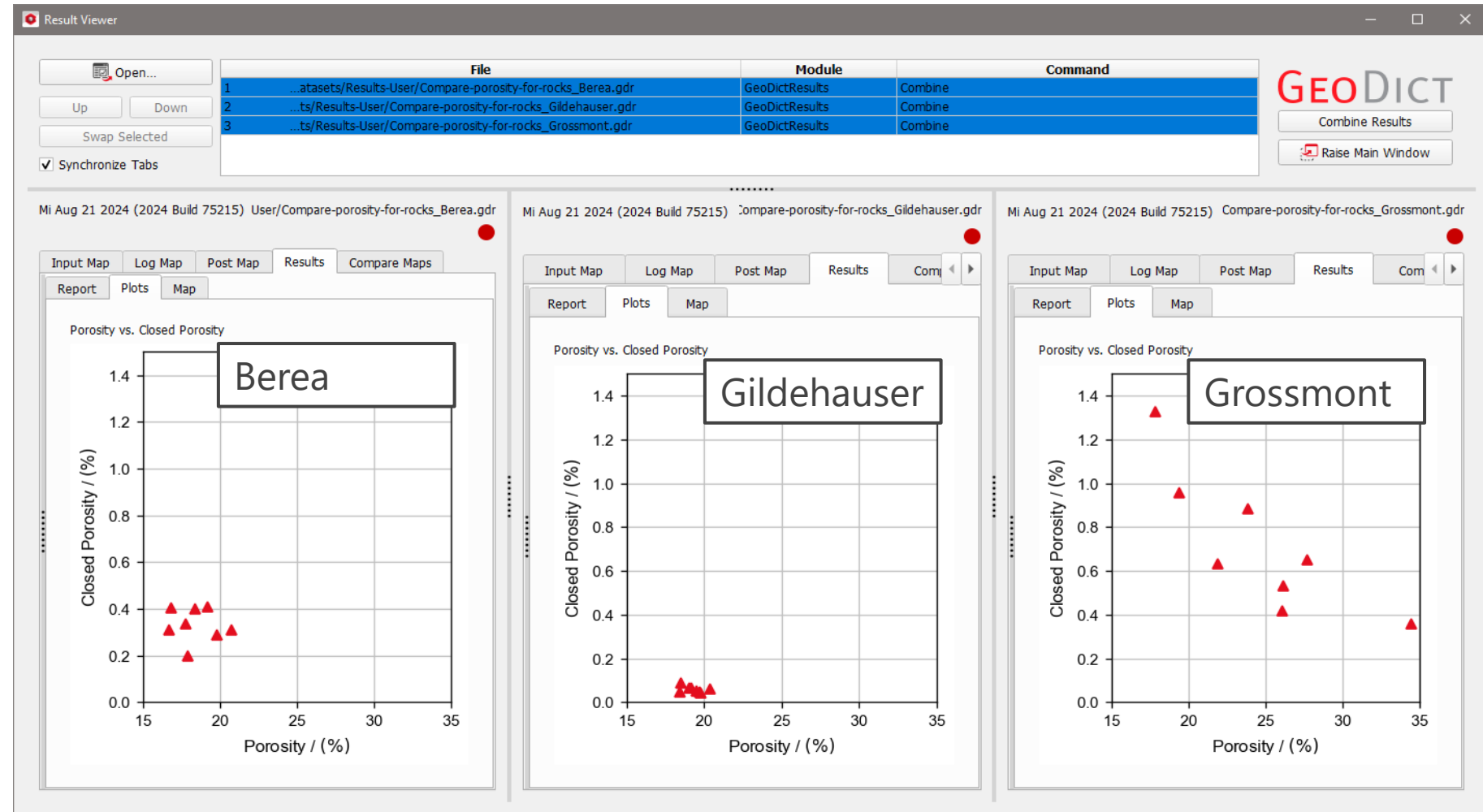
The results generated by the macro are opened in the Result Viewer automatically.



PART 3 – POROSITY RESULTS FOR EACH DIGITAL ROCK



Now we have the results for the three digital rocks separated. The same plot settings are applied

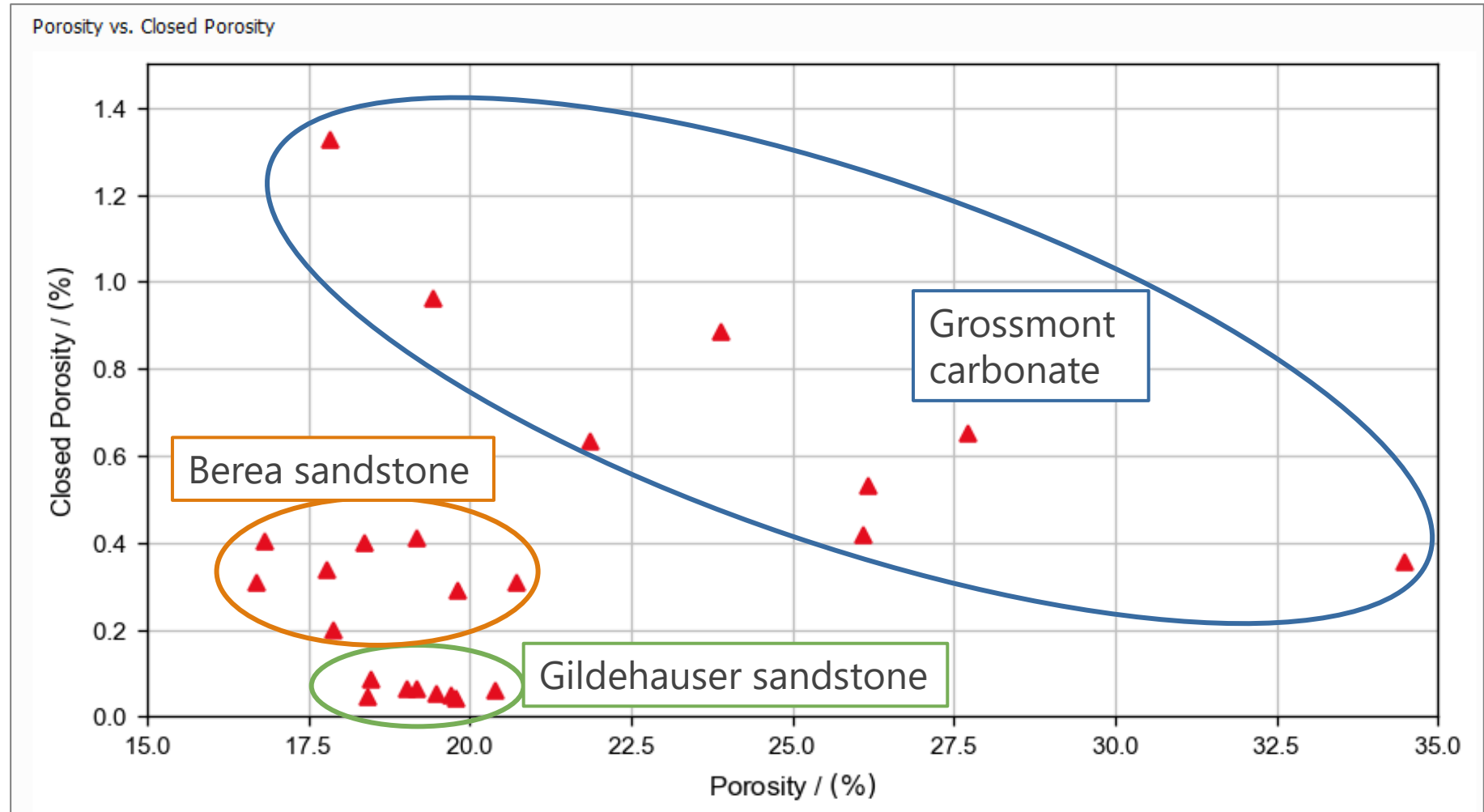


PART 3 – RESERVOIR ROCK CHARACTERIZATION

BASED ON POROSITY ANALYSIS

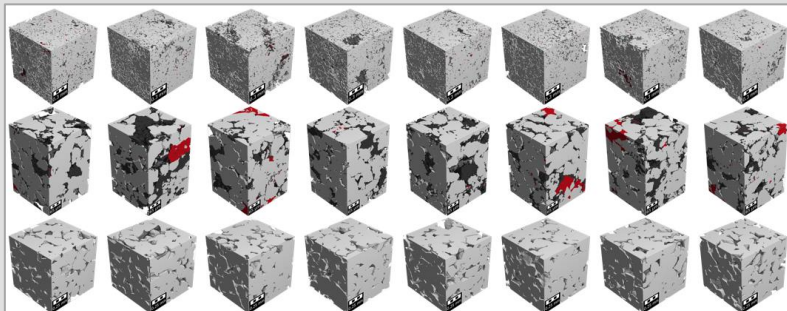


The digital rocks are characterized and grouped by Porosity vs. Closed Porosity ratio



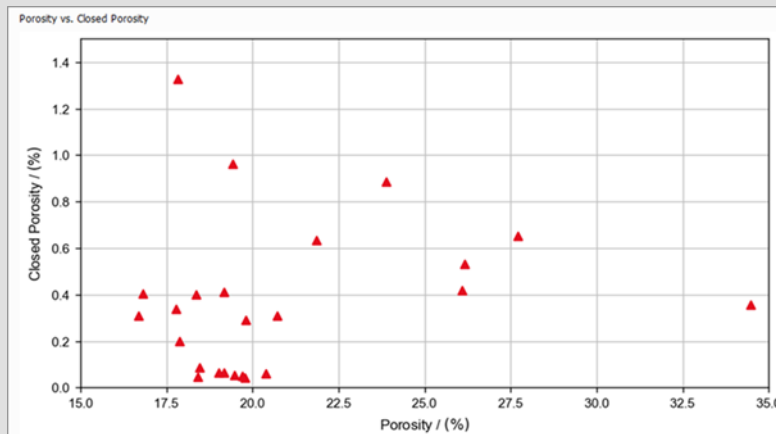
- 1 GeoDict automation possibilities and tutorial set-up
- 2 Record and edit a GeoDict macro to handle multiple datasets
- 3 Summary and outlook

WE CREATED A **GEO**PY SCRIPT TO LOAD STRUCTURES AND COMPUTE POROSITY



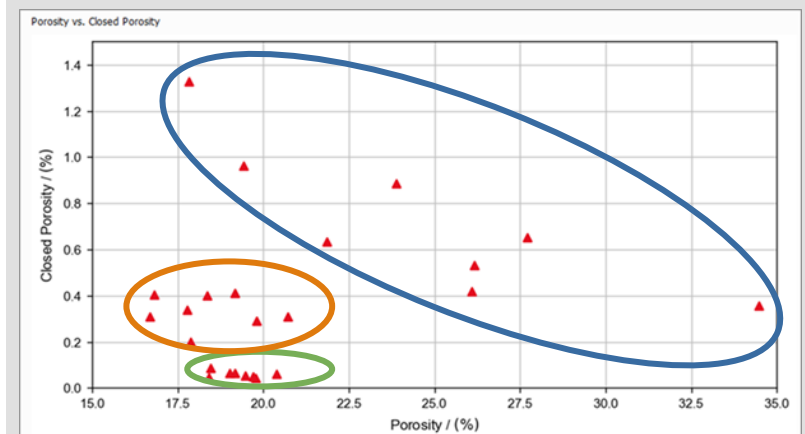
PART 1

WE COMBINED RESULTS INTO A SINGLE FILE AND CREATED A CUSTOM PLOT



PART 2

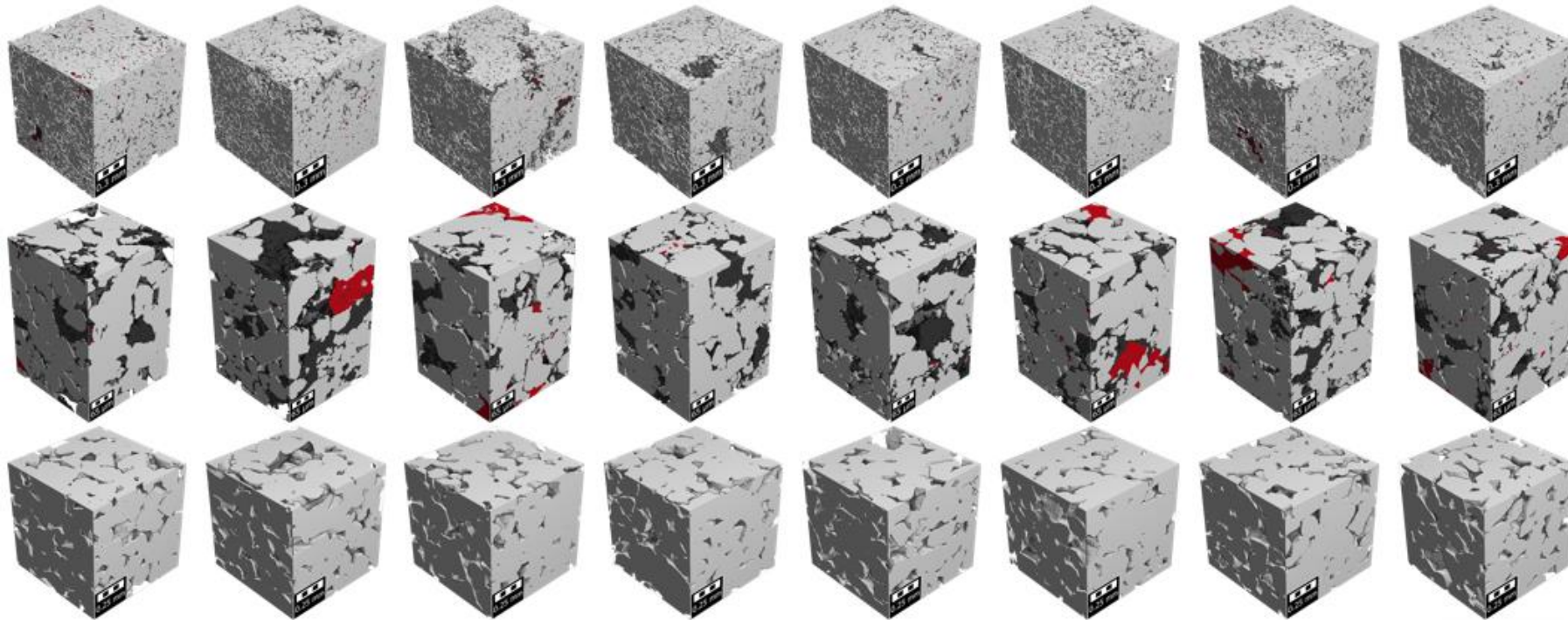
WE AUTOMATED POST-PROCESSING & FOUND CLUSTERS CHARACTERIZING 3 DIFFERENT DIGITAL ROCKS



PART 3

Automate the handling of multiple datasets for your own simulations.

As exercise: save images with the same visualization settings from a batch of structure files.




AUTOMATION

GEOPY – PYTHON INTERFACE

GEODict

Increase productivity by automating many workflows.

- Automate common/repetitive tasks
 - Record and playback **GeoDict** macros
- Perform parameter studies
 - Add variables to **GeoPy** macros
- Generate PowerPoint reports with **GeoPy**
 - Summarize & compare results
- Use entire range of  **Python** coding
 - Access any kind of **GeoDict** data
- Link your **GeoPy** scripts as GeoApps in the **GeoDict** menu bar

You can automate just about anything in **GeoDict**

