User Guide for the Humeral Medial Cropping Protocol: A Step-by-Step Manual

Content

How to prepare your Slicer environment

Step 1: Install Required Software

Note: It is recommended to restart 3D Slicer after installing these tools to ensure proper functionality. This step only needs to be done once.

Required for all functions:

3D Slicer – Download the latest version from the 3D Slicer website.

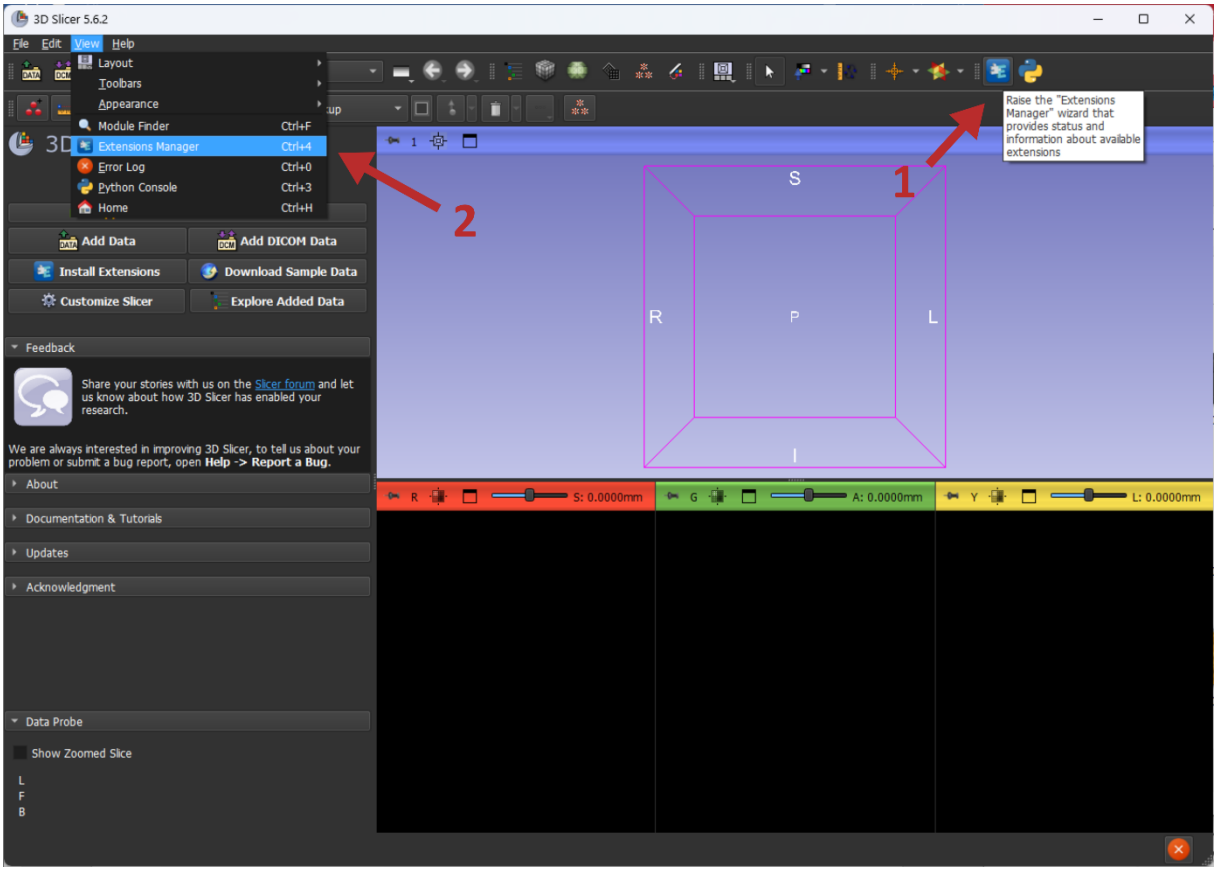
Required for the remesh procedure:

PyMeshLab – This will be automatically downloaded during script execution. Alternatively, it can be manually installed from the Python Package Index website.

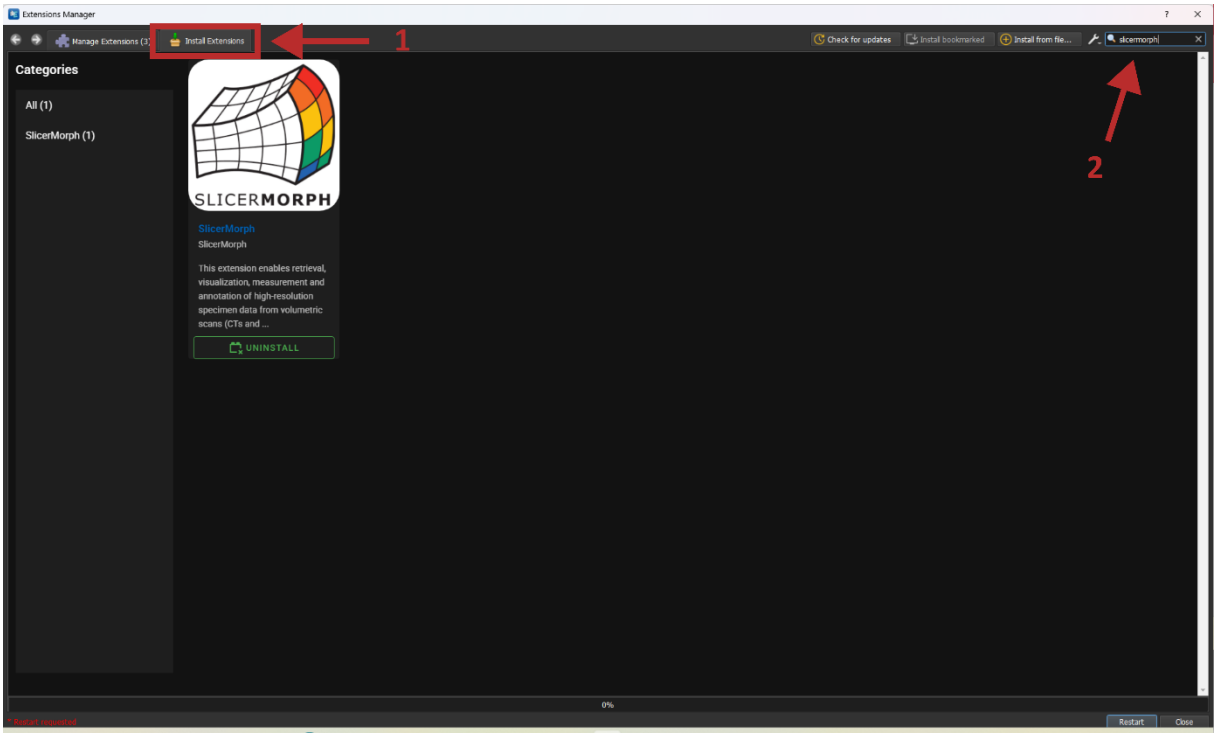
Required for the alignment procedure:

SlicerMorph Extension – Install via the Extension Manager in 3D Slicer.

- Access the Extension Manager either through the toolbar (1) or by navigating to **View → Extension Manager** in the menu (2).
- The following screenshot highlights both options:



- Under the tab “Install Extensions” (1), search for “SlicerMorph” (2) and install this extension.



For more details about SlicerMorph, visit the SlicerMorph website.

References

3D Slicer. (2024, September 9). 3D Slicer image computing platform | 3D Slicer. Retrieved September 16, 2024, from 3D Slicer is a free, open source software for visualization, processing, segmentation, registration, and analysis of medical, biomedical, and other 3D images and meshes; and planning and navigating image-guided procedures. website: https://www.slicer.org/

Muntoni, A., & Cignoni, P. (2024). *PyMeshLab: PyMeshLab v2023.12.post2*. Zenodo. doi: 10.5281/zenodo.13768931

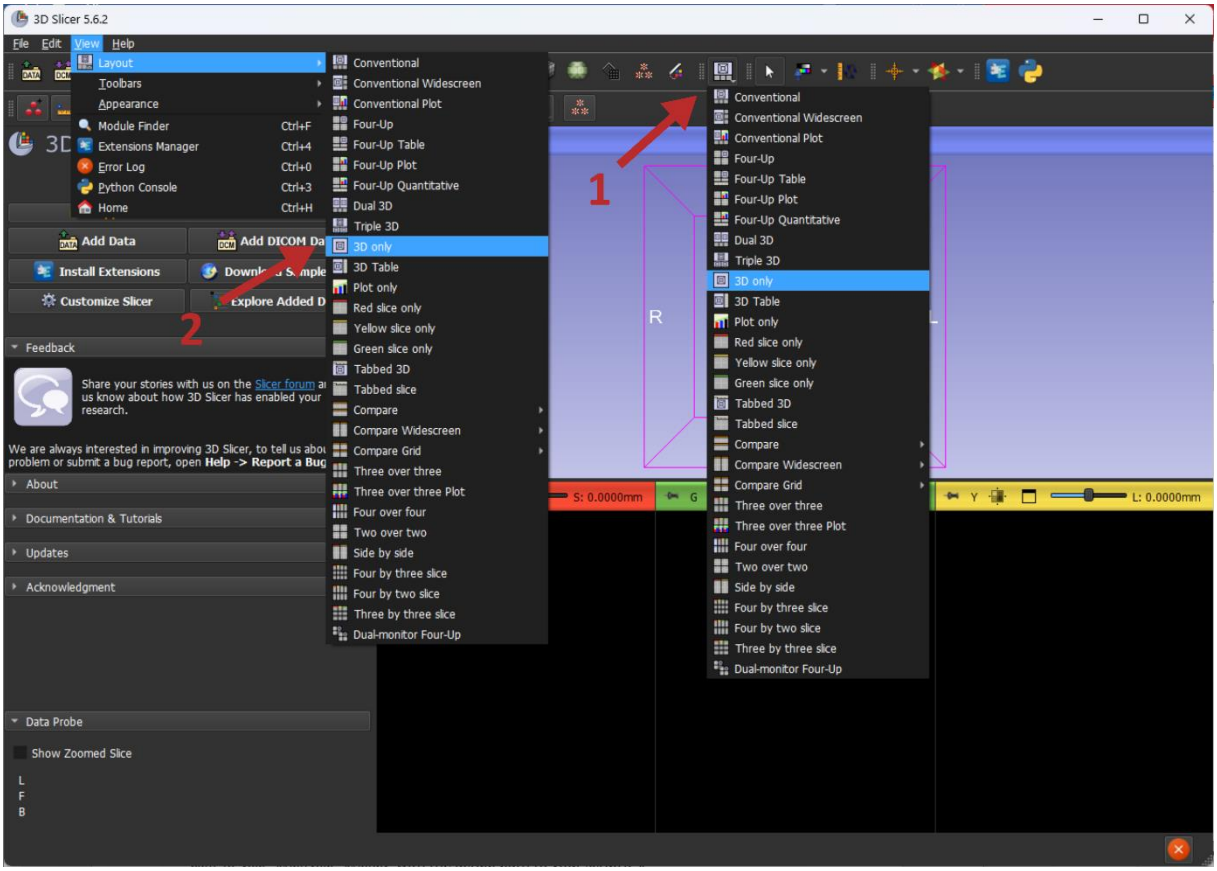
Porto A, Rolfe S, Maga AM. 2021. ALPACA: A fast and accurate computer vision approach for automated landmarking of three-dimensional biological structures. *Methods in Ecology and Evolution* **12**: 2129–2144. DOI: 10.1111/2041-210X.13689

Rolfe S, Pieper S, Porto A, Diamond K, Winchester J, Shan S, Kirveslahti H, Boyer D, Summers A, Maga AM. 2021. SlicerMorph: An open and extensible platform to retrieve, visualize and analyse 3D morphology. *Methods in Ecology and Evolution* **12**: 1816–1825. DOI: 10.1111/2041-210X.13669

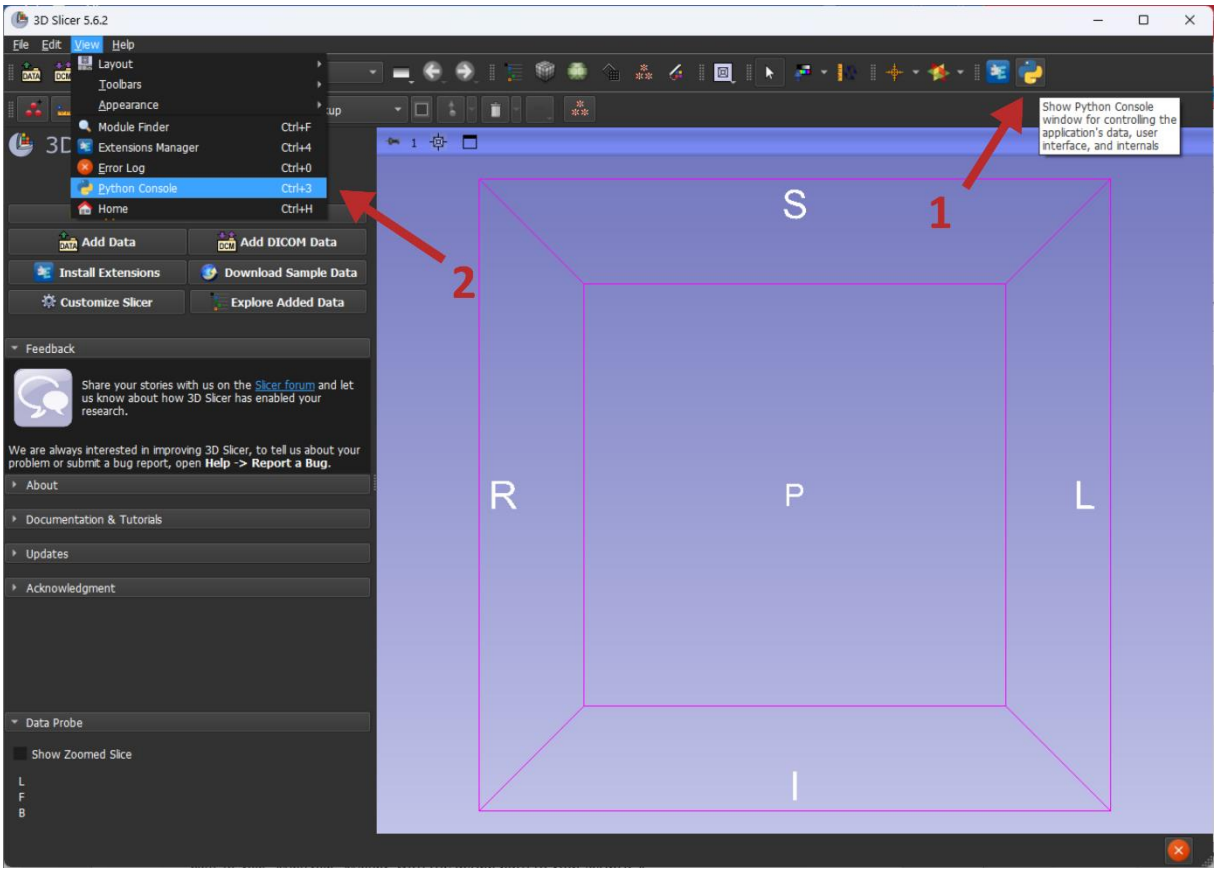
Step 2: Set Up Your 3D Slicer Environment

- **Open 3D Slicer.**
- By default, 3D Slicer uses the **"Conventional" layout**, which includes a **3D view** and three slice views (**red, green, and yellow**).

  Since the following functions require **only the 3D view**, you may prefer switching to the **"3D only" layout** for better visibility. You can change the layout by clicking on the **"Layout selection" icon** in the toolbar (1), or navigating to **View** → **Layout** → **3D only** in the menu bar (2). The following screenshot highlights both options:



- **Open the Python Console**, as it will be needed for the next steps. You can do this by clicking the **Python logo** in the toolbar (1), or by navigating to **View** → **Python Console** in the menu bar (2). The following screenshot highlights both options:



Step 3: Add Functions to 3D Slicer

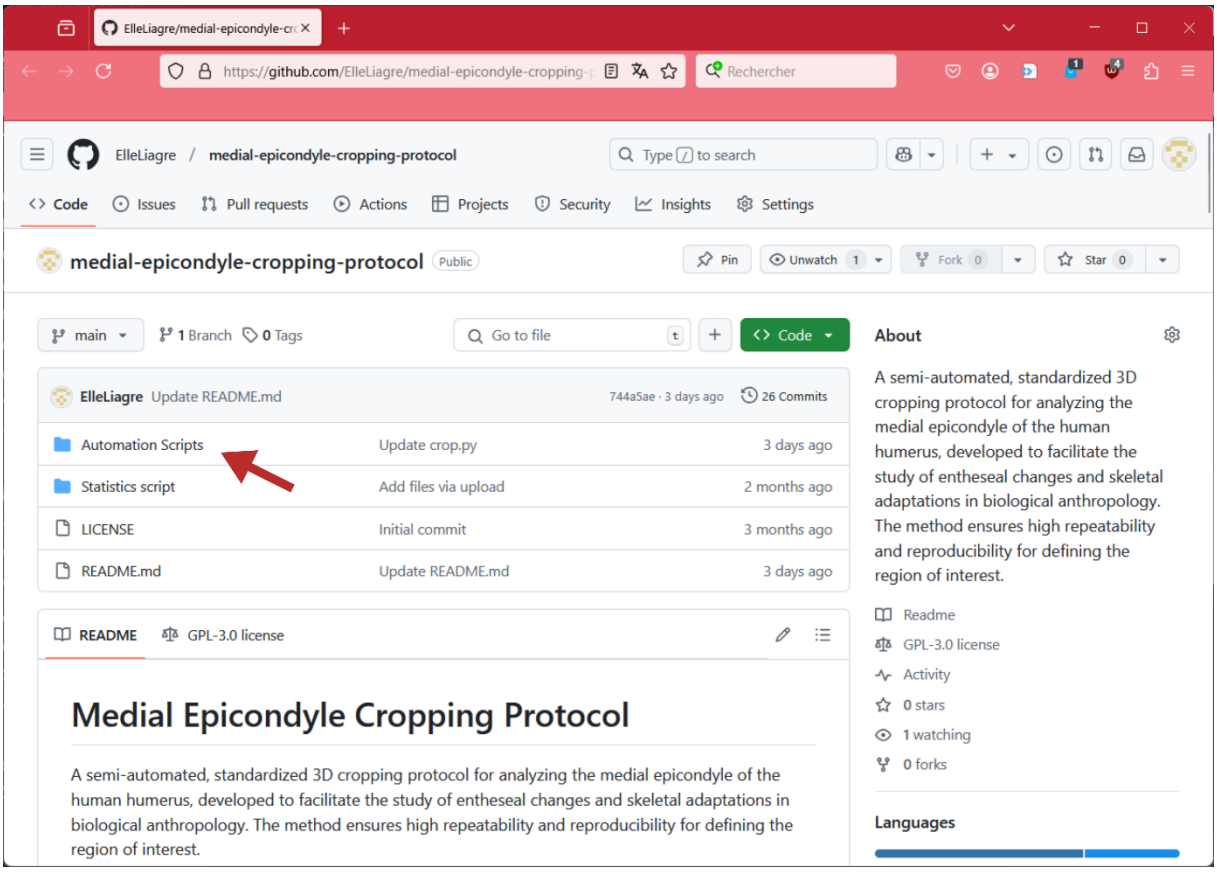
- Open 3D Slicer and go to the **Python Console** (see Step 2).
- There are two ways to add a script to 3D Slicer:
    - Method 1: Copy and paste the Python script into the console
    - Method 2: Load a python script from a file (*.py)
        - by running (replace 'path_to_your_script/your_script.py' with the actual path to your script):

```
exec(open(r'path_to_your_script/your_script.py').read())
```

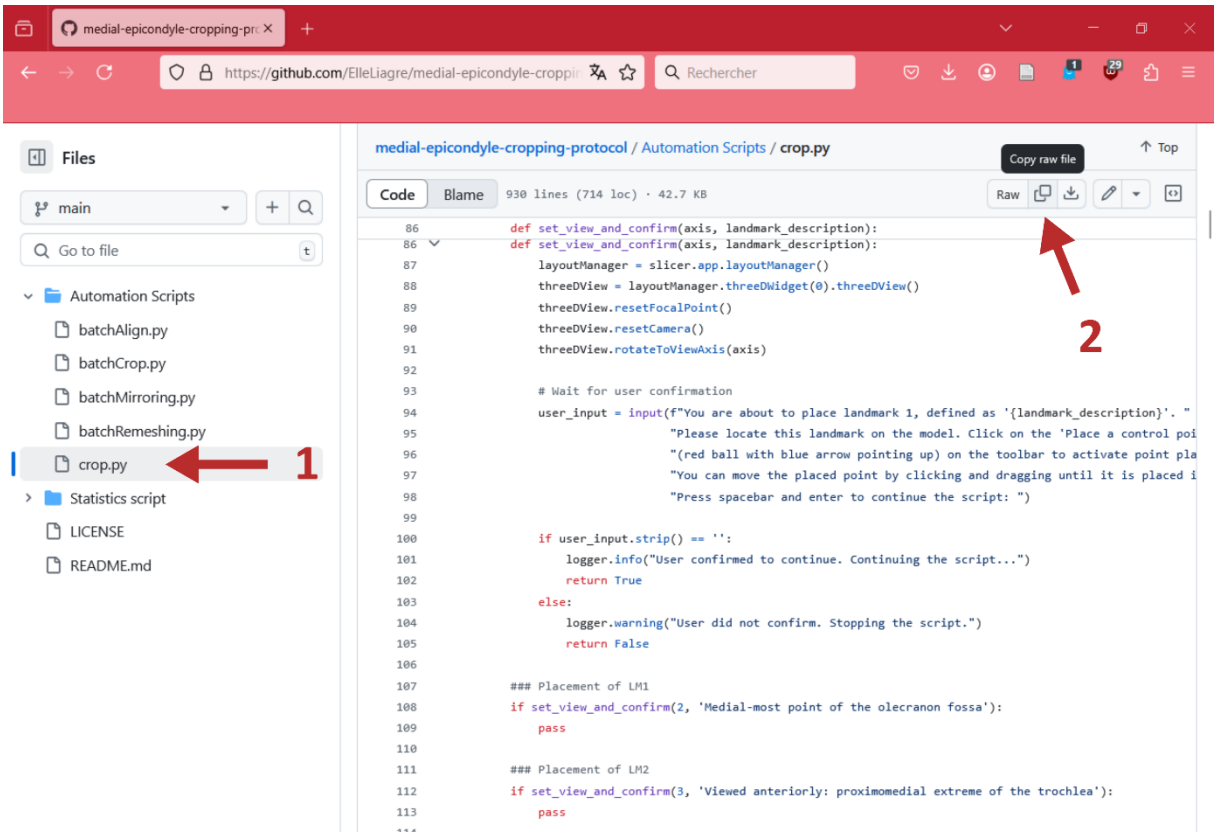
        - Press **Enter** to execute the script.

Demonstration of method 1

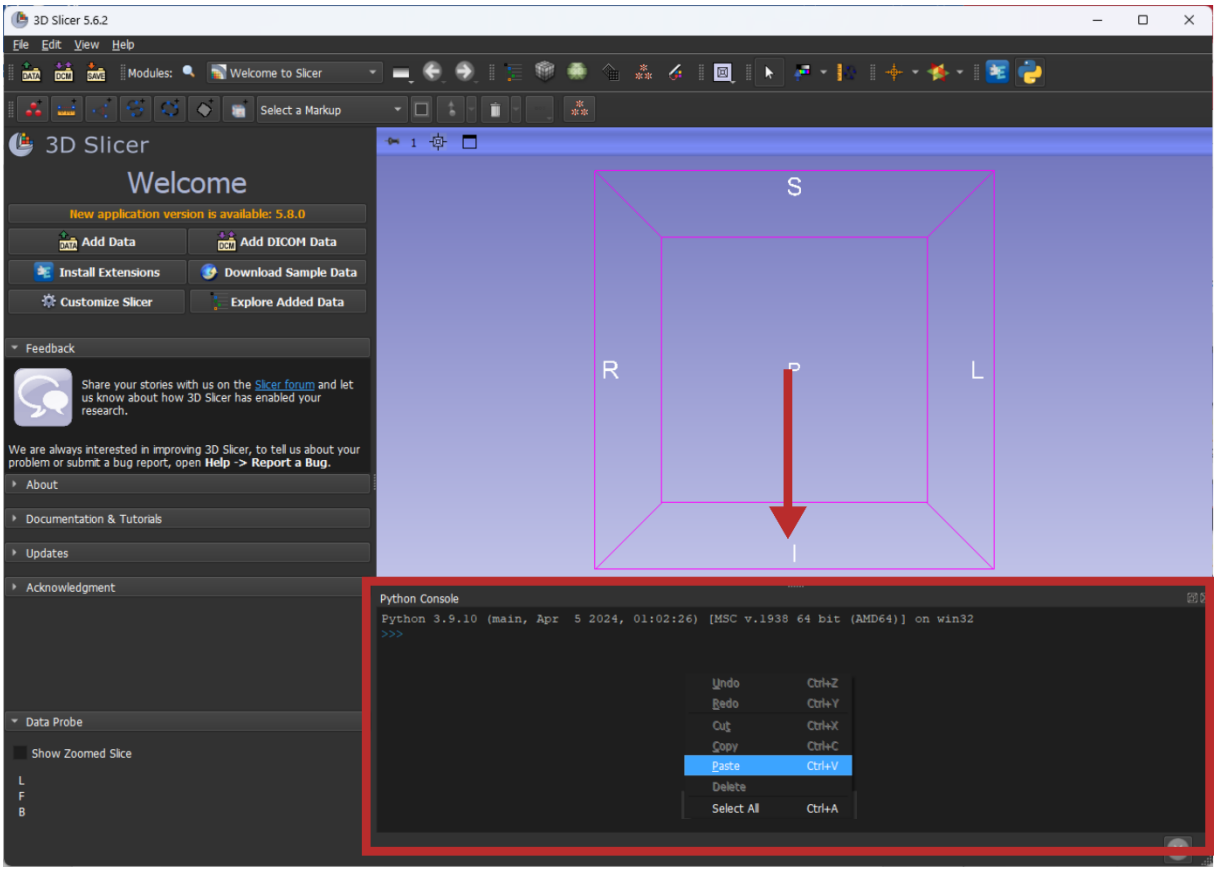
Locate the script you want to use from the **GitHub repository** under **automation scripts**.



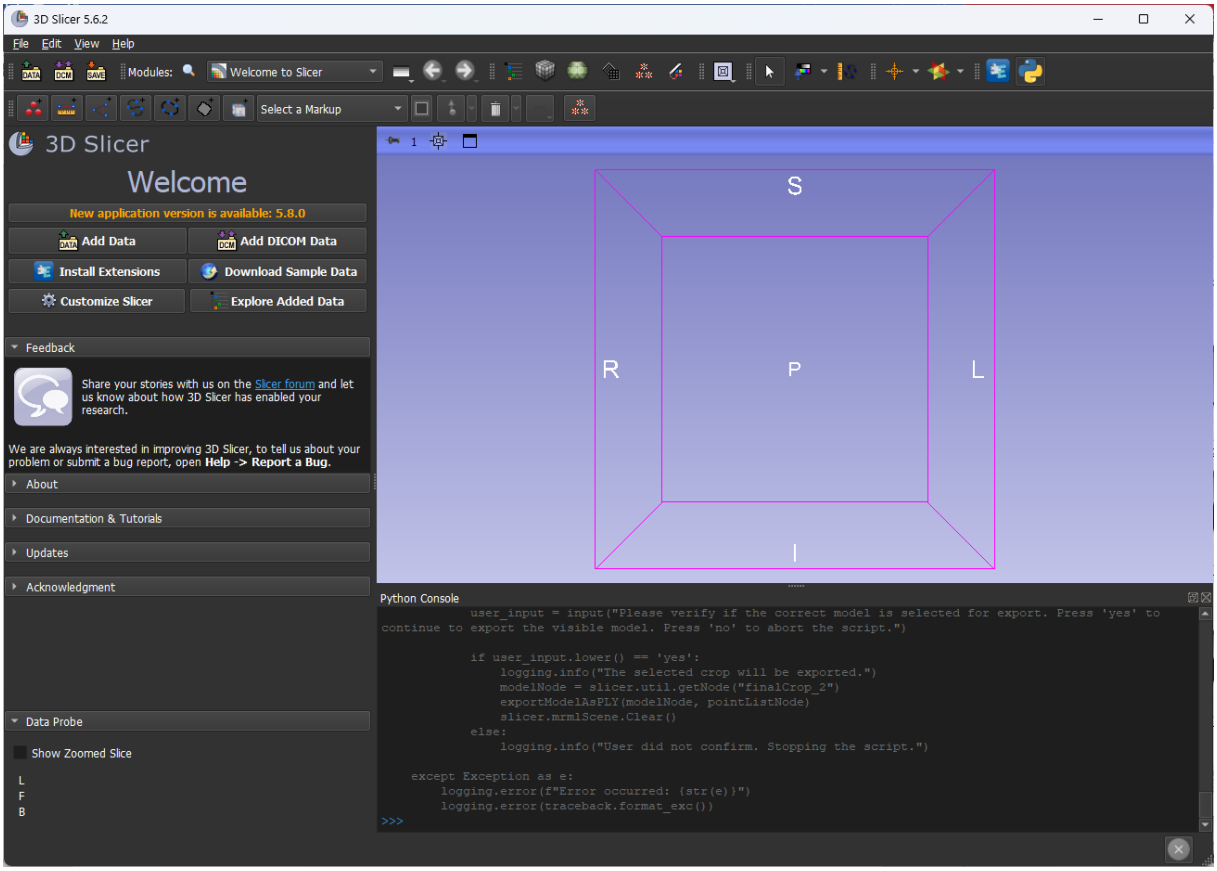
Open the desired script, e.g. “crop.py” (1), and copy it (2).



Return to **3D Slicer**, right-click in the **Python Console**, and select **Paste**.

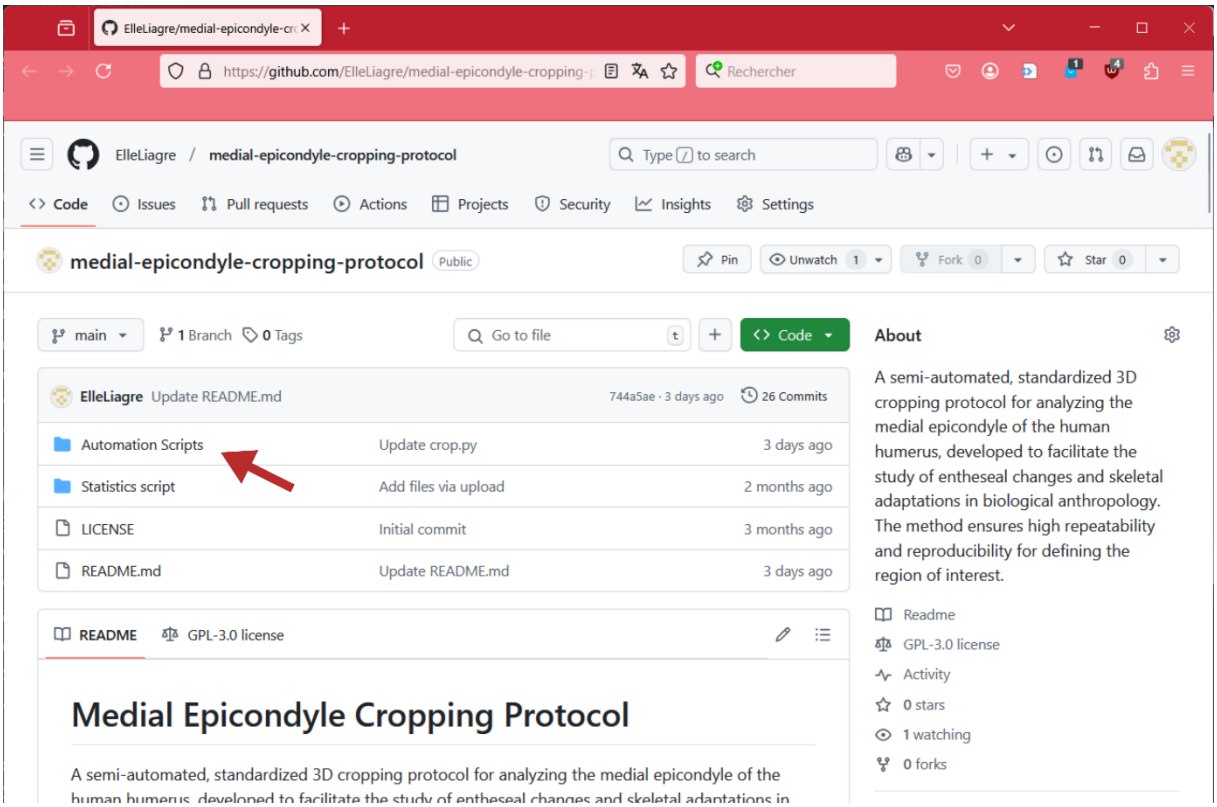


Expected outcome after adding it to the Python Console.

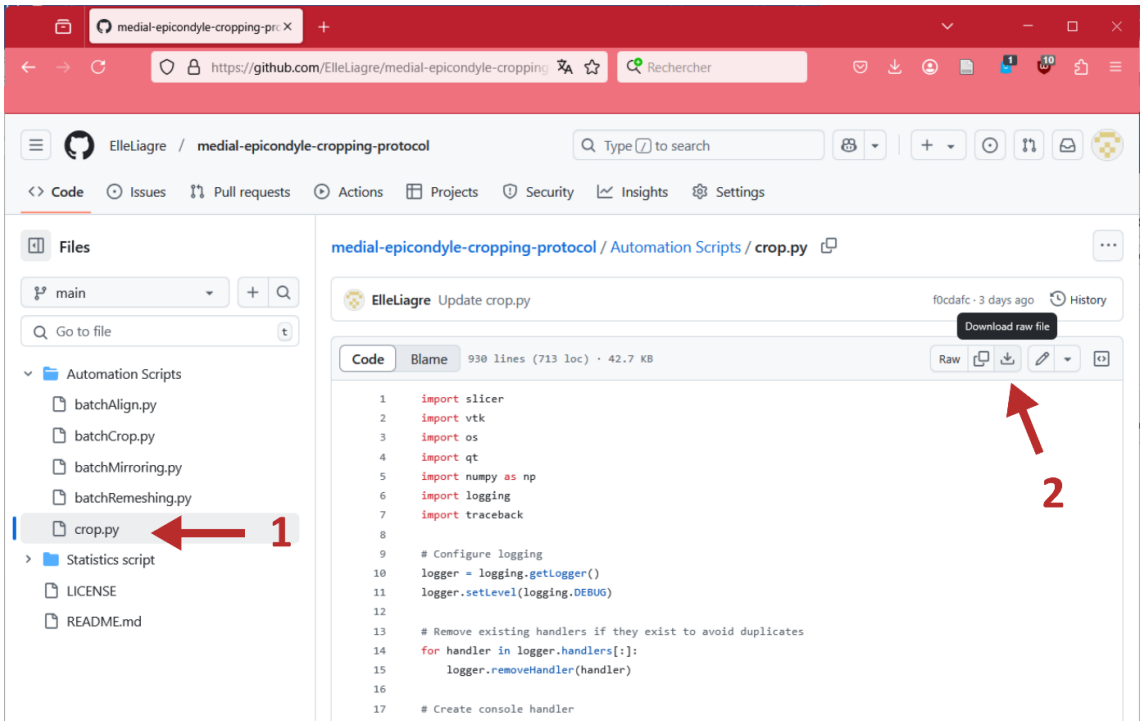


<u>Demonstration of method 2</u>

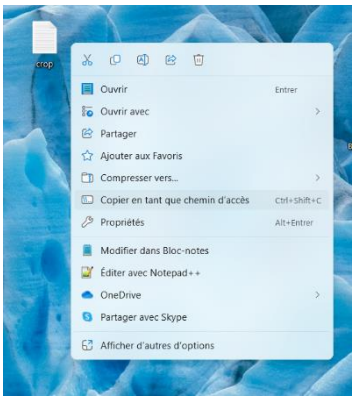
Locate the script you want to use from the **GitHub repository** under **automation scripts**.



Open the desired script, e.g. “crop.py” (1), and download it (2).



Find the downloaded file on your computer, **right-click** on it, and select **"Copy as Path"** (in the screenshot, this appears as *"Copier en tant que chemin d'accès"* in French).



Return to **3D Slicer**, click inside the **Python Console**, and type the following command:

```
exec(open(r'
```

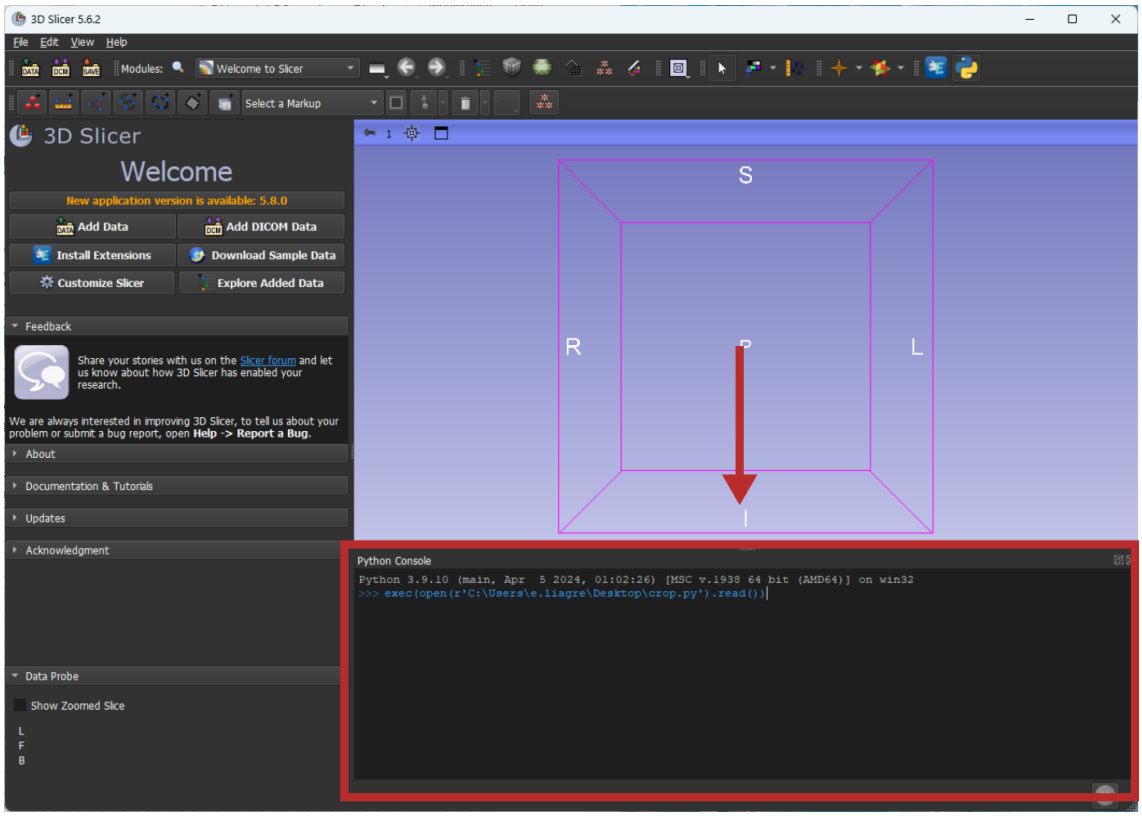
Right-click and **paste** the file path you just copied. For example, the pasted path might look like this:

```
"C:\Users\e.liagre\Desktop\crop.py"
```

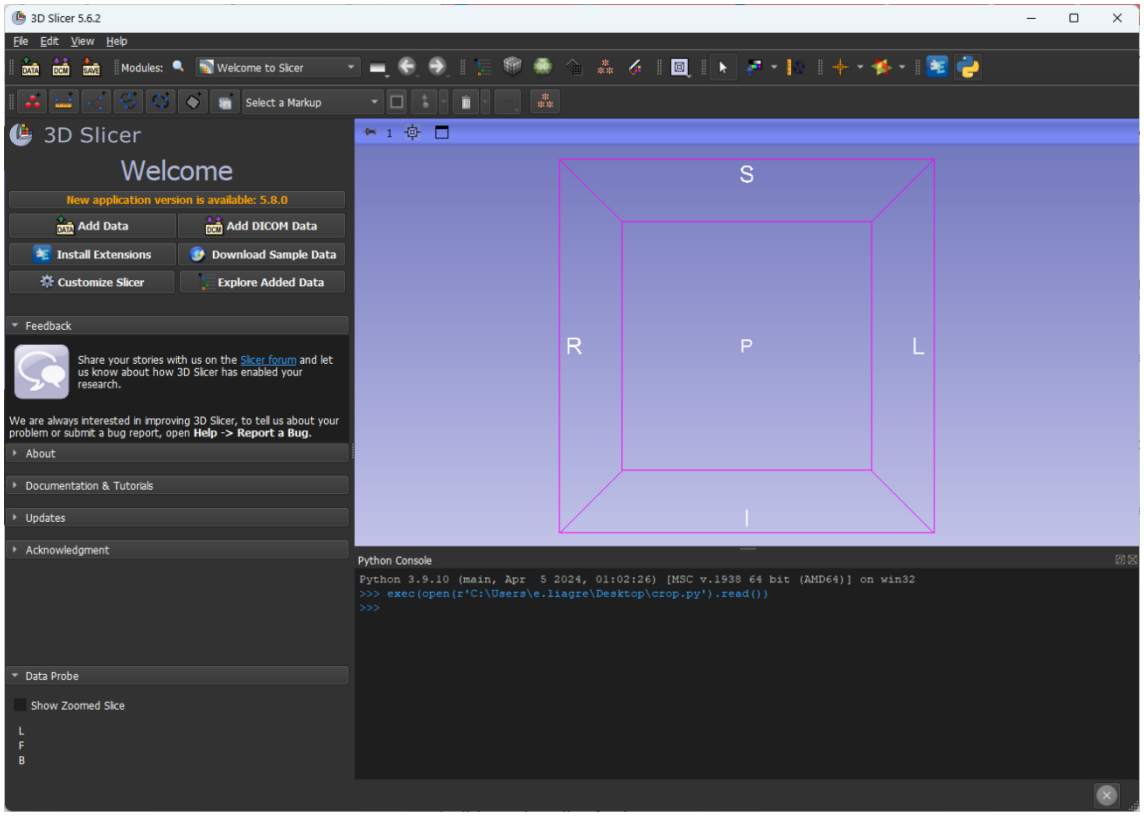
Important: Remove the quotation marks (“”) around the path. Then, complete the command by adding:

```
').read())
```

Press **Enter** to execute the script.



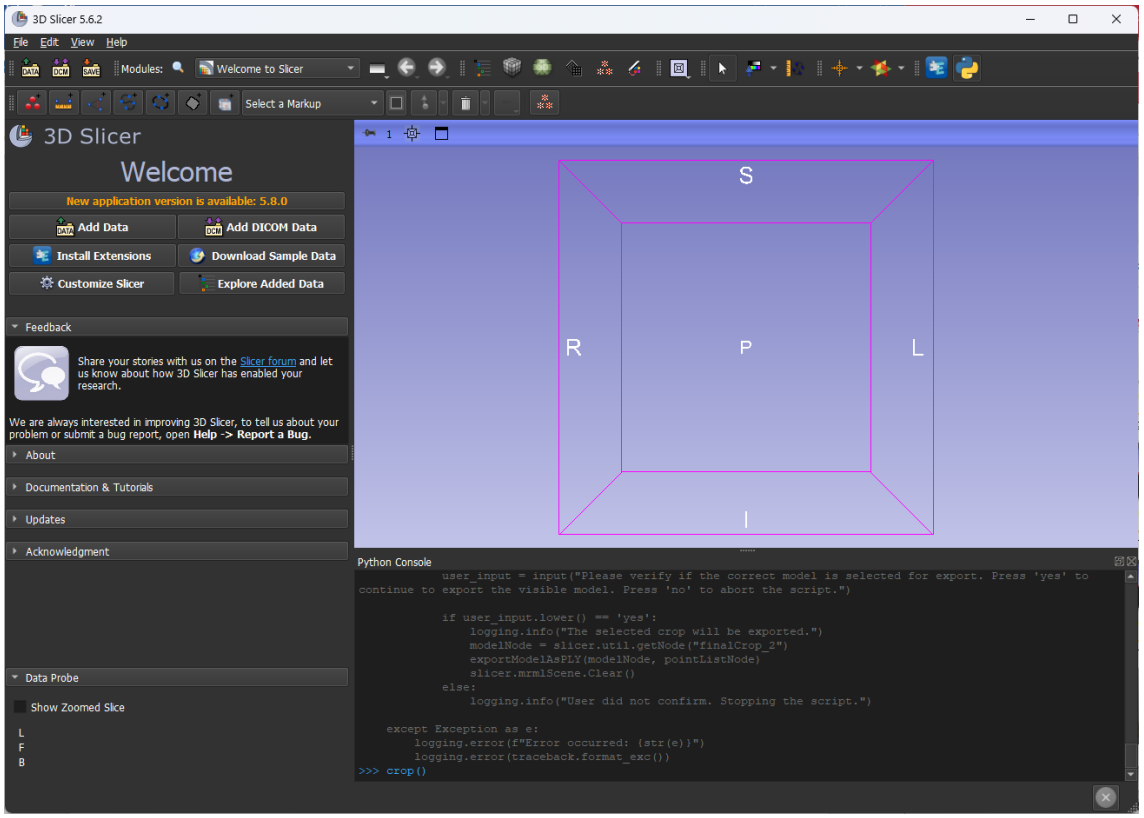
Expected outcome after script execution.



Step 4: Execute the Function

To execute a function, enter the function call in the Python Console and press **Enter**. For example, to execute the crop function, type: “crop()” and press **Enter** (See the screenshot below for reference).

Detailed explanations of each function and their purposes can be found in the next chapter 'The Functions'.



The Functions

batchRemeshing.py

Description

This function allows you to remesh all 3D models within a folder according to a specified mesh resolution. The remeshing process runs in batch mode, processing multiple models simultaneously.

Pre-requisites

- **Required software:** 3D Slicer installed (See “How to prepare your Slicer environment, Step 1”)
- **Required script:** “batchRemeshing” script added to 3D Slicer (See “How to prepare your Slicer environment, Step 2 to 4”)

Note: The first time the script is executed, it checks if PyMeshlab is installed. If not, it will attempt to install it. You may need to restart 3D Slicer and re-add the script for it to function correctly.

Usage

To use the batch remeshing function, run the following command in the Python Console:

```
batchRemeshing(r'/path/to/your/folder', res = 22)
```

Arguments

`/path/to/your/folder`	Path to the folder containing 3D models. If omitted, a file dialog will prompt you to select a folder. You can obtain the folder path using the same method described in "How to Prepare Your Slicer Environment, Step 3, Method 2."
`res`	Mesh resolution, defined as faces per mm². The default value is 22.

Details

The script creates a new folder within the original directory to store the remeshed models.

All models within the folder will be standardized to the same mesh resolution, defined by the number of faces per mm².

The remeshing uses two PyMeshlab filters (v2023.12) (Muntoni & Cignoni, 2024):

- Remeshing: Isotropic Explicit Remeshing
- Simplification: Quadric Edge Collapse Decimation

For more information on these filters and their parameters, see “Manual Application”.

Example use of calls

```
batchRemeshing()
```

Runs the script with default settings and prompts for a folder selection.

```
batchRemeshing(res = 10)
```

Runs the script with a **mesh resolution of 10** and prompts for folder selection.

```
batchRemeshing(r'/path/to/your/folder')
```

Runs the script on a **specified folder** (`/path/to/your/folder`), using the default mesh resolution.

Manual Application

To perform manual remeshing using MeshLab (v2022.02) (Cignoni et al., 2008):

- Open the 3D model in Meshlab.
- Apply the **"Remeshing: Isotropic Explicit Remeshing"** filter with a target length and maximal surface distance of **0.3 mm** to achieve uniform face distribution.
- Use the **"Simplification: Quadric Edge Collapse Decimation"** filter to adjust the mesh resolution.
    - Set the **Quality Threshold** to 1.
    - Enable the options: "Preserve boundary", "Preserve Normal", "Preserve Topology", and "Optimal position of simplified vertices."
- To calculate the target number of faces, multiply the desired resolution (e.g., 22 faces/mm²) by the total surface area of the mesh. The total surface area of the mesh can be obtained by using the "**Compute Geometric Measures**" filter.

References

Cignoni P, Callieri M, Corsini M, Dellepiane M, Ganovelli F, Ranzuglia G. 2008. MeshLab: an Open-Source Mesh Processing Tool. Scarano V, De Chiara R, and Erra U (eds). *Europgraphics Italian Chapter Conference* : 8.

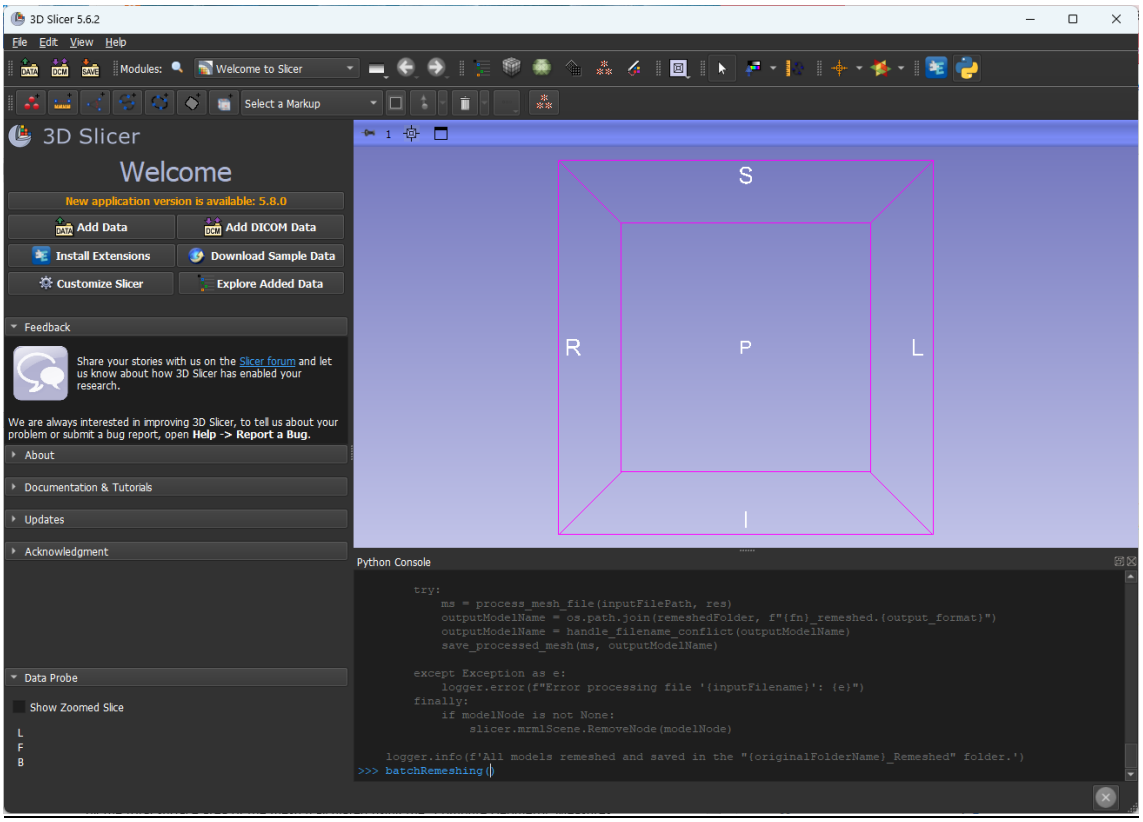
Muntoni A, Cignoni P. 2024. PyMeshLab: PyMeshLab v2023.12.post2 DOI: 10.5281/zenodo.13768931

Visual Guide to Executing the Function

Call the function –In the Python Console, type:

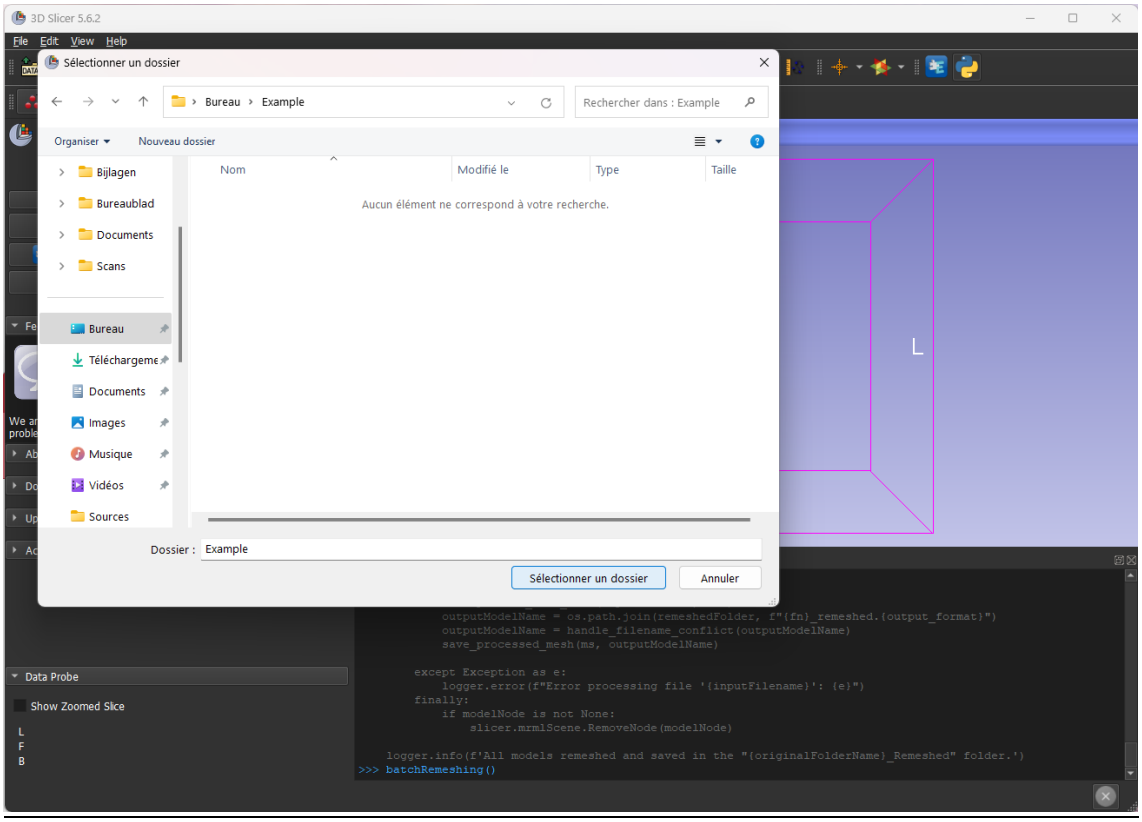
```
batchRemeshing()
```

and press **Enter**.

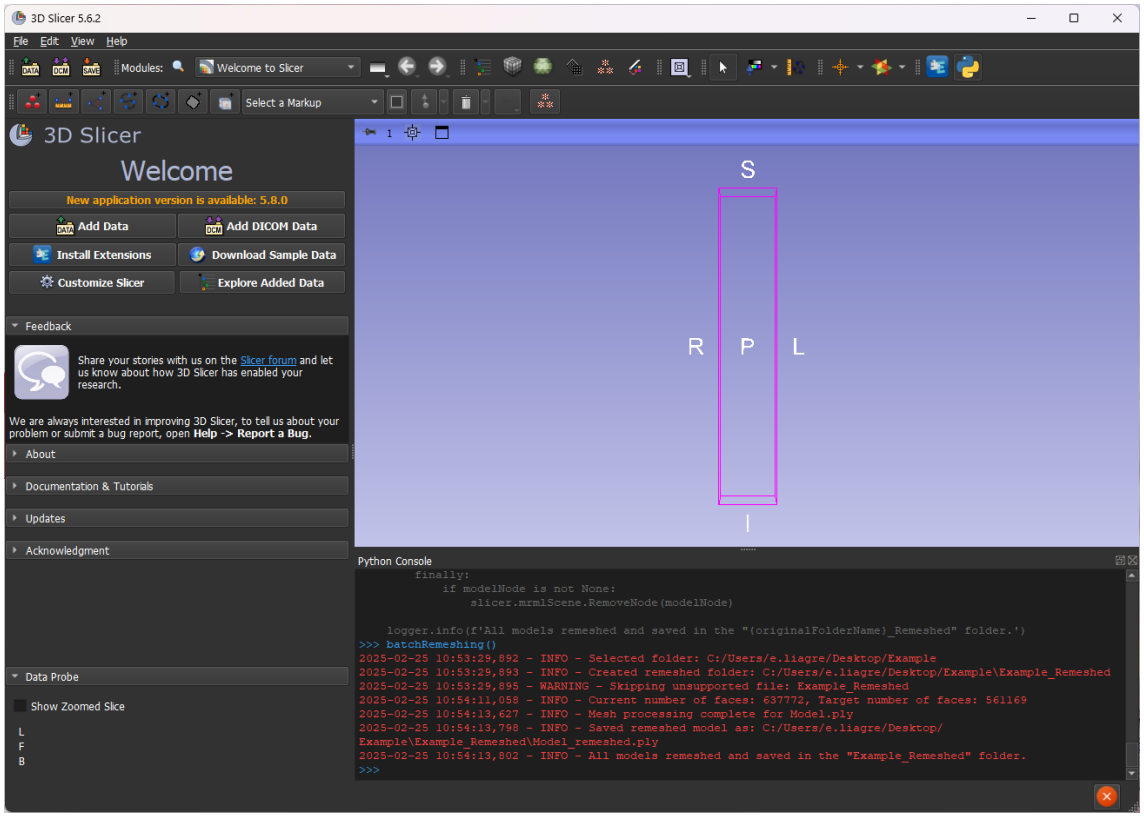


Select a folder (if no path is specified) – If you have not provided a folder path, a window will appear asking you to select the folder containing the models you want to remesh.

- Navigate to the desired folder.
- Click "Select Folder" (appears as "Sélectionner un dossier" in French in the screenshot).



Remeshing Process – The script will create a new folder to store the remeshed models, process all models in the selected folder and save them to the new folder. The process is complete when the Python Console displays the message: **"All models remeshed and saved in the 'Example_Remeshed' folder"** *(where "Example" is replaced by your folder's name)*, and when the console prompt (>>>) appears below the messages, indicating the script has finished running.



batchMirroring.py

Description

This batch function mirrors all 3D models in a specified folder. If no folder path is provided, a file dialog will prompt you to select one. The mirrored models are saved in a new folder within the original directory.

Pre-requisites

- **Required software:** 3D Slicer installed (See “How to prepare your Slicer environment, Step 1”)
- **Required script:** “batchMirroring” script added to 3D Slicer (See “How to prepare your Slicer environment, Step 2 to 4”)

Usage

To use the batch mirroring function, run the following command in the Python Console:

```
batchMirroring(r'/path/to/your/folder')
```

Arguments

`/path/to/your/folder`	Path to the folder containing 3D models. If omitted, a file dialog will prompt you to select a folder. You can obtain the folder path using the same method described in "How to Prepare Your Slicer Environment, Step 3, Method 2."

Details

The script creates a new folder within the original directory to store the mirrored models.

All models within the folder will be mirrored along the x-axis, using 3D Slicer’s Surface Toolbox.

Example use of call

```
batchMirroring()
```

Runs the script and prompts for a folder selection.

```
batchMirroring(r'/path/to/your/folder')
```

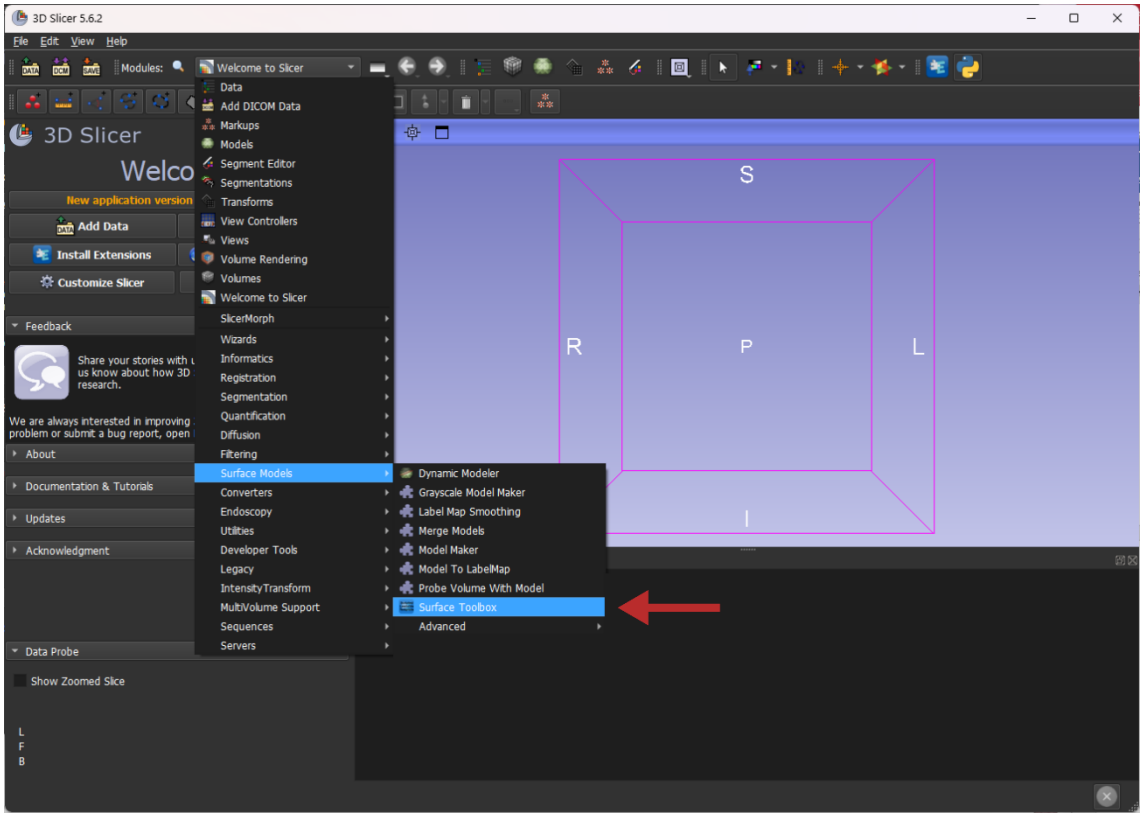
Runs the script on a **specified folder** (`/path/to/your/folder`).

Manual Application

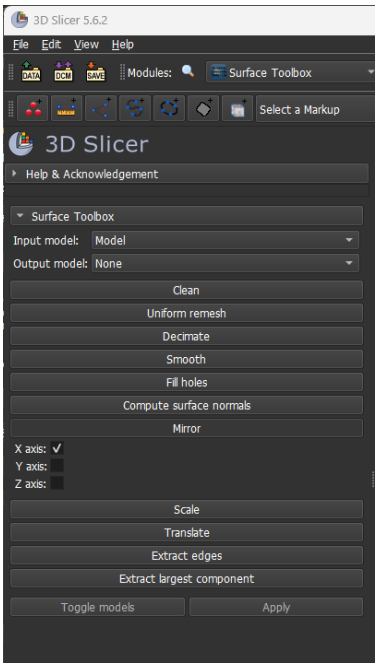
3D Slicer's **Surface Toolbox** module provides a manual way to mirror models along the X, Y, or Z axis.

Steps:

1. **Import your model** in 3D Slicer
    - Drag and drop the model into the Slicer interface, or
    - Go to **File > Add Data**, then select your model.
2. **Navigate to the Surface Toolbox module** (*see screenshot below*).



In this module, select your model as the **Input Model**. You can then choose how the **Output Model** should be created. To mirror the model, click on the **Mirror** function and select the axis along which you want to perform the mirroring (e.g., X-axis, Y-axis, or Z-axis). After selecting the desired axis, press **Apply** to execute the transformation.

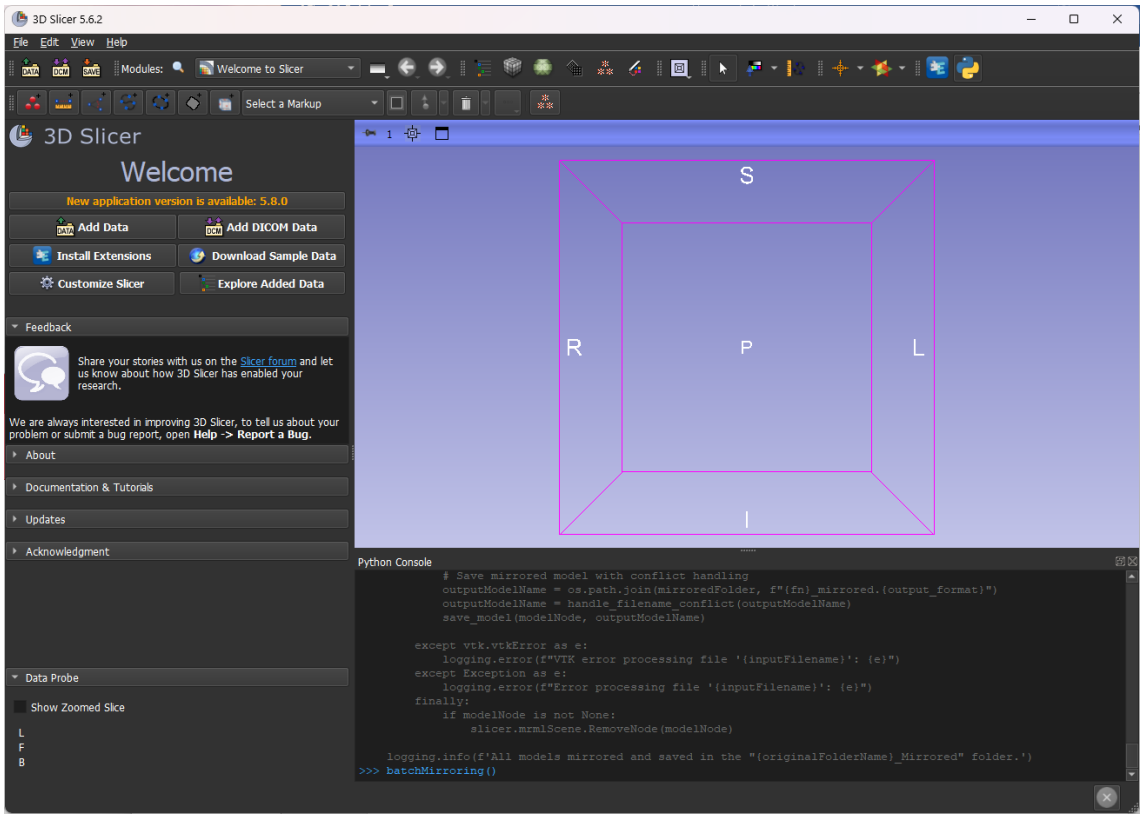


Visual Guide to Executing the Function

Call the function – In the Python Console, type:

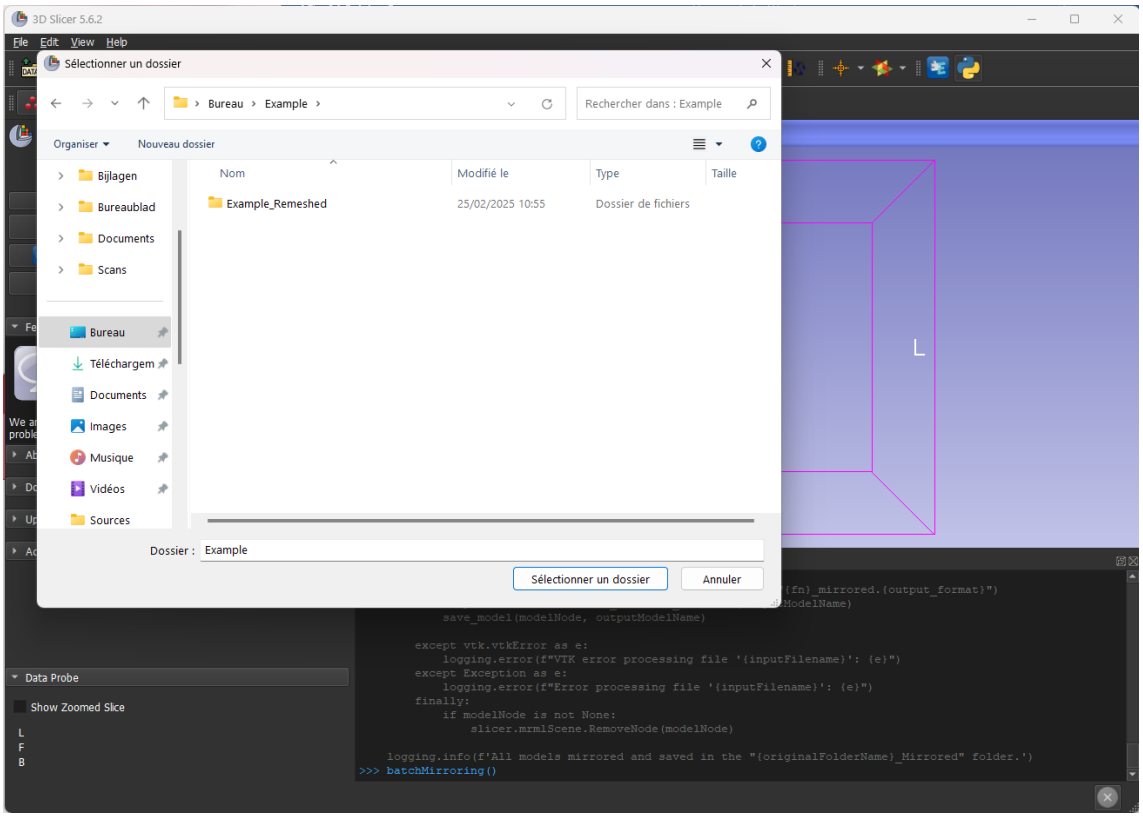
```
batchMirroring()
```

and press **Enter**.

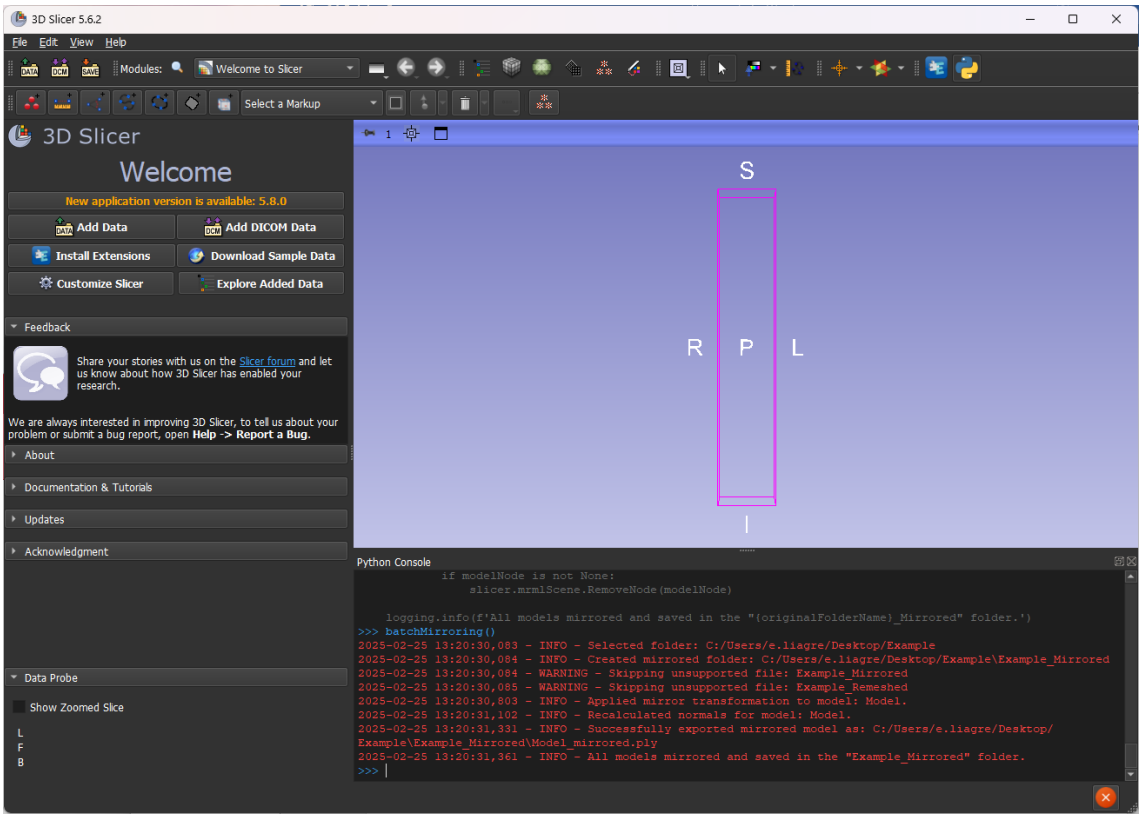


Select a folder (if no path is specified) – If you have not provided a folder path, a window will appear asking you to select the folder containing the models you want to remesh.

- Navigate to the desired folder.
- Click "Select Folder" (appears as "Sélectionner un dossier" in French in the screenshot).



Mirroring Process – The script will create a new folder to store the mirrored models, and will process all models in the selected folder. The process is complete when the Python Console displays the message: **"All models mirrored and saved in the 'Example_Mirrored' folder"** *(where "Example" is replaced by your folder's name)*, and when the console prompt (>>>) appears below the messages, indicating the script has finished running.



batchAlign.py

Description

This function allows you to align all 3D models within a folder to a reference model. The alignment process runs in batch mode, processing multiple models simultaneously.

Pre-requisites

- **Required software:** 3D Slicer and the "SlicerMorph" extension installed (See "How to prepare your Slicer environment, Step 1")
- **Required script:** "batchAlign" script added to 3D Slicer (See "How to prepare your Slicer environment, Step 2 to 4")
- **Loaded reference model:** Before running the function, load the reference model into 3D Slicer and name it **"target"**. A reference model is provided (see Data Availability), which is anatomically positioned, with the longitudinal axis of the bone aligned along the Z-axis, medio-distal alignment along the X-axis, and antero-posterior alignment along the Y-axis. Alternatively, you can use your own reference model.
    - **Importing the reference model** in 3D Slicer
        - Drag and drop the model into the Slicer interface, or
        - Go to **File > Add Data**, then select your model.
    - **Rename** the model (if necessary):
        - Navigate to the Data module.
        - Double left-click or right-click and select **Rename**, then name the model "target".

Note: When executed, the script checks if *SlicerMorph* is installed. If not, it will prompt you to install it. After installation, you may need to restart 3D Slicer and re-add the script and reference model.

Usage

To use the batch alignment function, run the following command in the Python Console:

```
batchAlign(r'/path/to/your/folder')
```

Arguments

`/path/to/your/folder`	Path to the folder containing 3D models. If omitted, a file dialog will prompt you to select a folder. You can obtain the folder path using the same method described in "How to Prepare Your Slicer Environment, Step 3, Method 2."

Details

The script creates a new folder within the original directory to store the aligned models.

All models in the folder are aligned to the same target model, using the *FastModelAlign* module in 3D Slicer, which is part of the SlicerMorph-extension (Porto et al., 2021; Rolfe et al., 2021). The module

aligns two 3D shapes by determining a transformation that correctly registers a source point cloud or image to a target point cloud or image.

The script uses the default settings of the FastModelAlign module, excluding object scaling. The alignment procedure iteratively adjusts point density until it meets the following quality thresholds (fitness threshold of 0.95; RMSE threshold of 3.4). The script will attempt alignment up to six times, progressively increasing the density threshold.

Important: Due to high morphological variation, not all alignments may be successful. Please verify the models' orientation after running the script. If necessary, perform alignment manually (see further).

Example use of call

```
batchAlign()
```

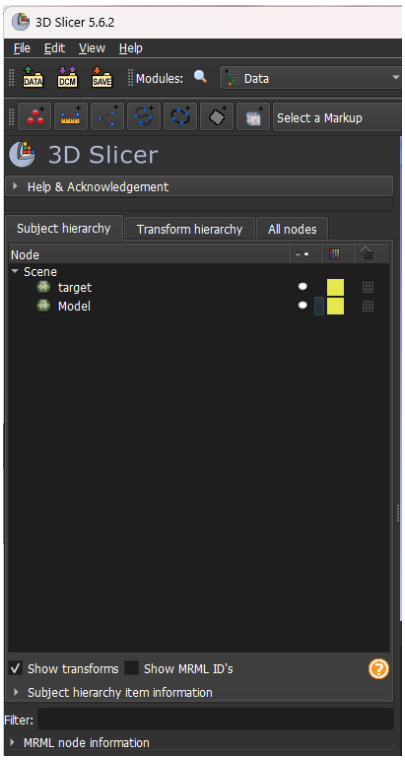
Runs the script and prompts for a folder selection.

```
batchAlign(r'/path/to/your/folder')
```

Runs the script on a **specified folder** (`/path/to/your/folder`).

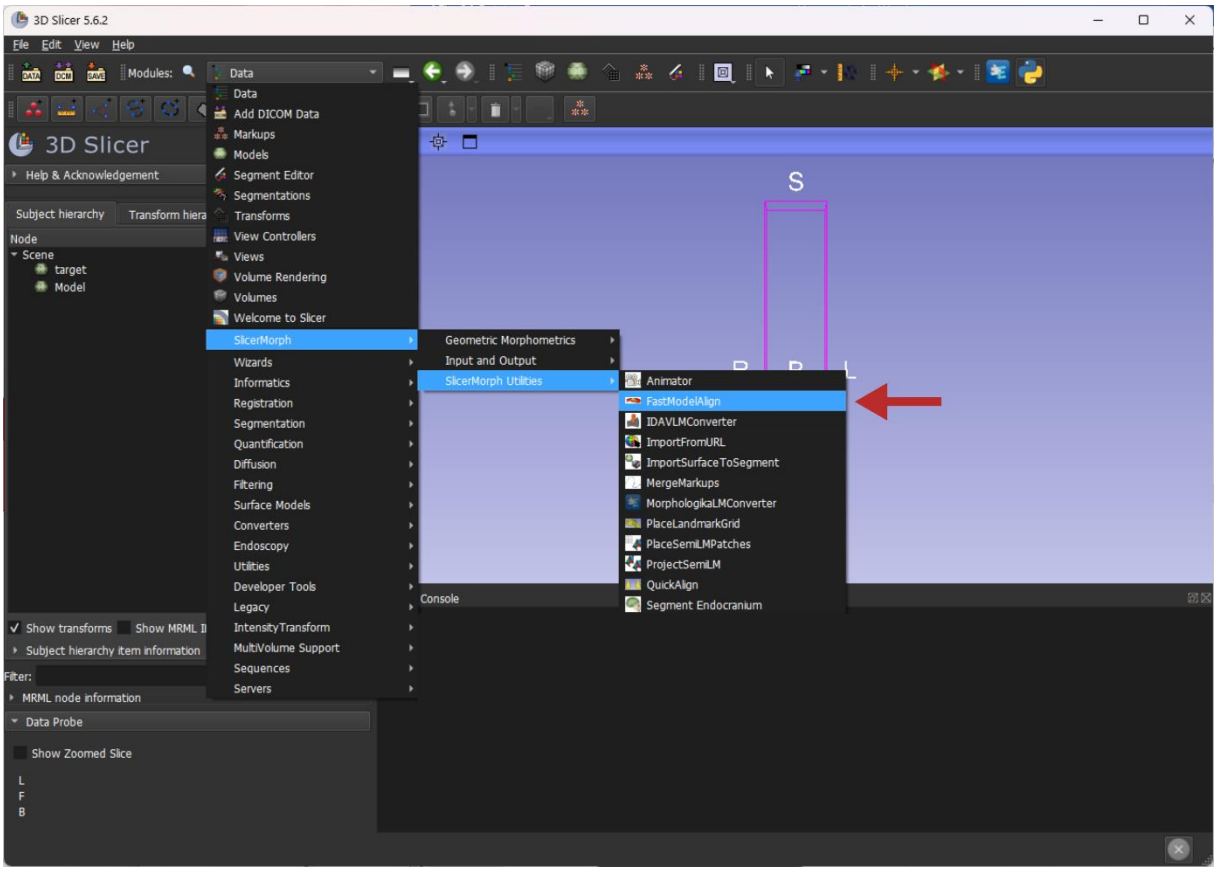
Manual Application

The FastModelAlign module, available through the **SlicerMorph** extension in 3D Slicer, allows for manual alignment of models to a reference model.



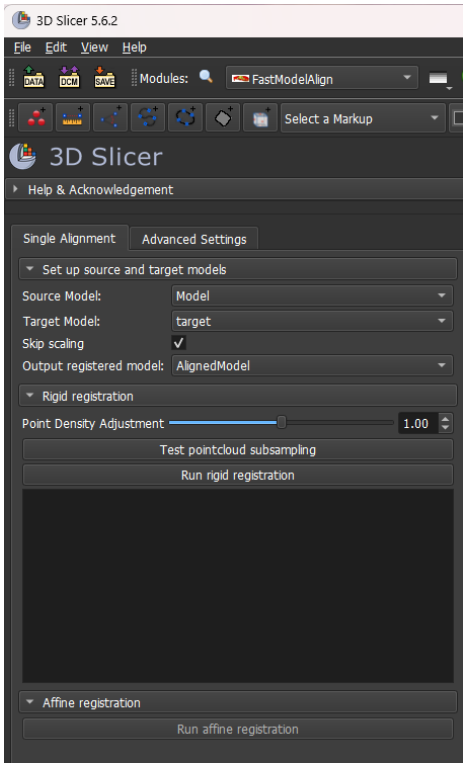
Steps:

1. Import your reference and to be aligned model in 3D Slicer
    - Drag and drop the models into the Slicer interface, or
    - Go to **File > Add Data**, then select your models.
    - **Note**: In the screenschot, both the reference model ("target") and the model to be aligned ("Model") are loaded.
2. **Navigate to the FastModelAlign module** (*see screenshot below*).



Inside the module, select the model you want to align as the **Source Model** and set the reference model as the **Target Model**. If you do not want the source model to be scaled to match the target model, check the "Skip scaling" option.

To perform the alignment, click **"Run rigid registration"** to apply a rigid transformation that aligns the source model to the target model. If further refinement is needed, you can also click **"Run affine registration"**, which applies additional transformations to improve the alignment, but seems to also perform scaling. Detailed instructions on how to use this module can be found in the [FastModelAlign tutorial].



References

Porto A, Rolfe S, Maga AM. 2021. ALPACA: A fast and accurate computer vision approach for automated landmarking of three-dimensional biological structures. *Methods in Ecology and Evolution* **12**: 2129–2144. DOI: 10.1111/2041-210X.13689

Rolfe S, Pieper S, Porto A, Diamond K, Winchester J, Shan S, Kirveslahti H, Boyer D, Summers A, Maga AM. 2021. SlicerMorph: An open and extensible platform to retrieve, visualize and analyse 3D morphology. *Methods in Ecology and Evolution* **12**: 1816–1825. DOI: 10.1111/2041-210X.13669

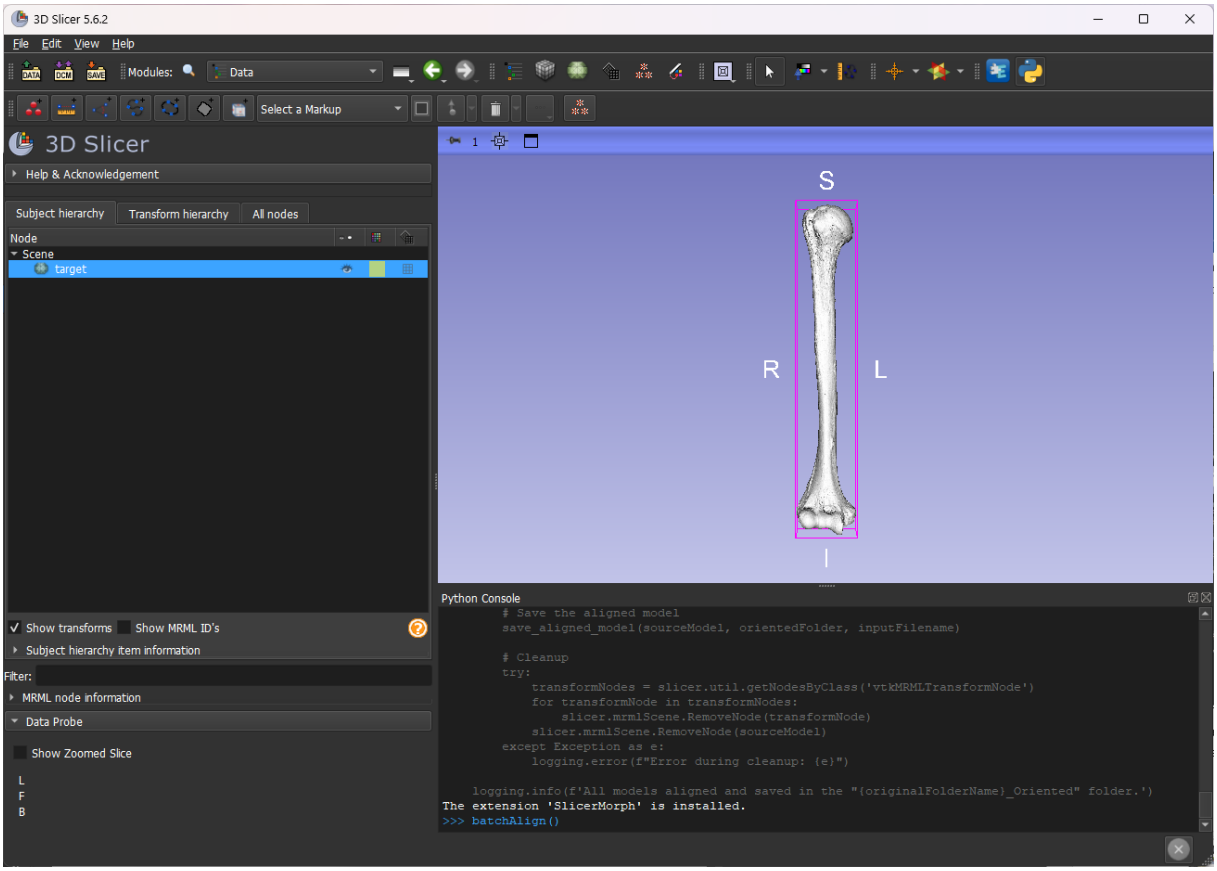
Visual Guide to Executing the Function

Make sure to have the reference model loaded as "target".

Call the function –In the Python Console, type:

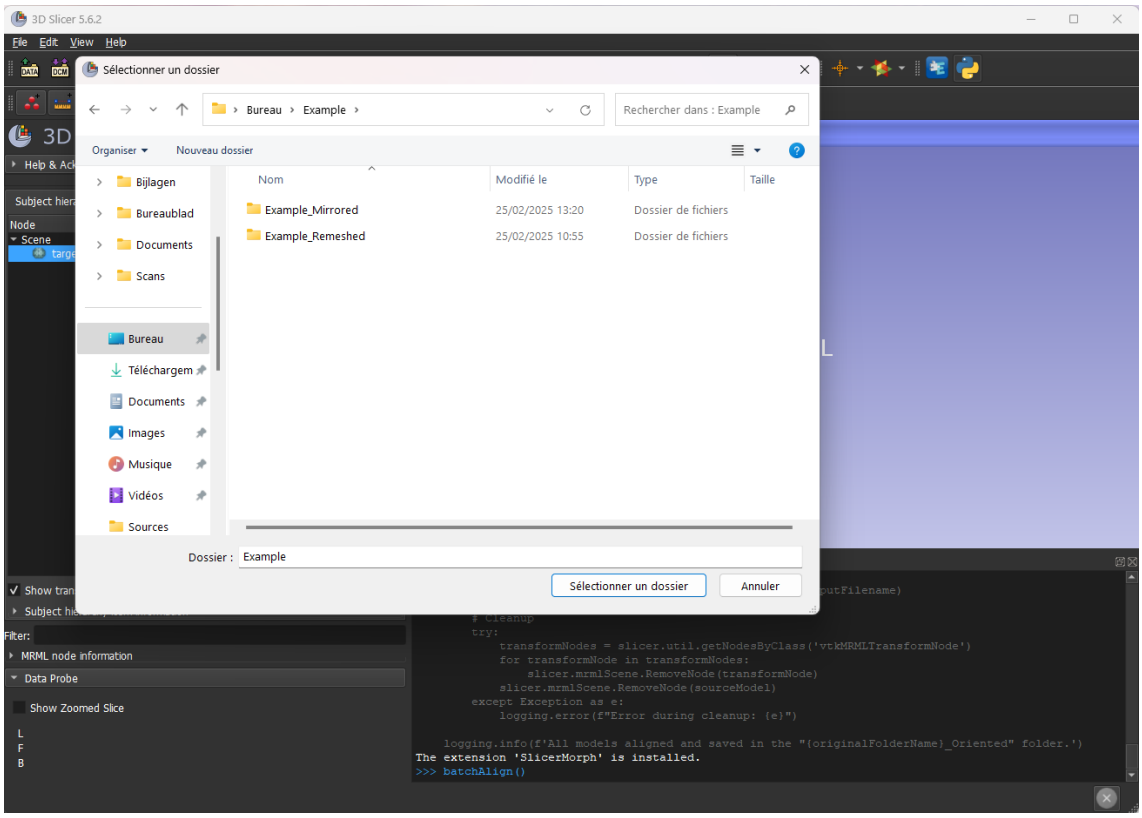
```
batchAlign()
```

and press **Enter**.

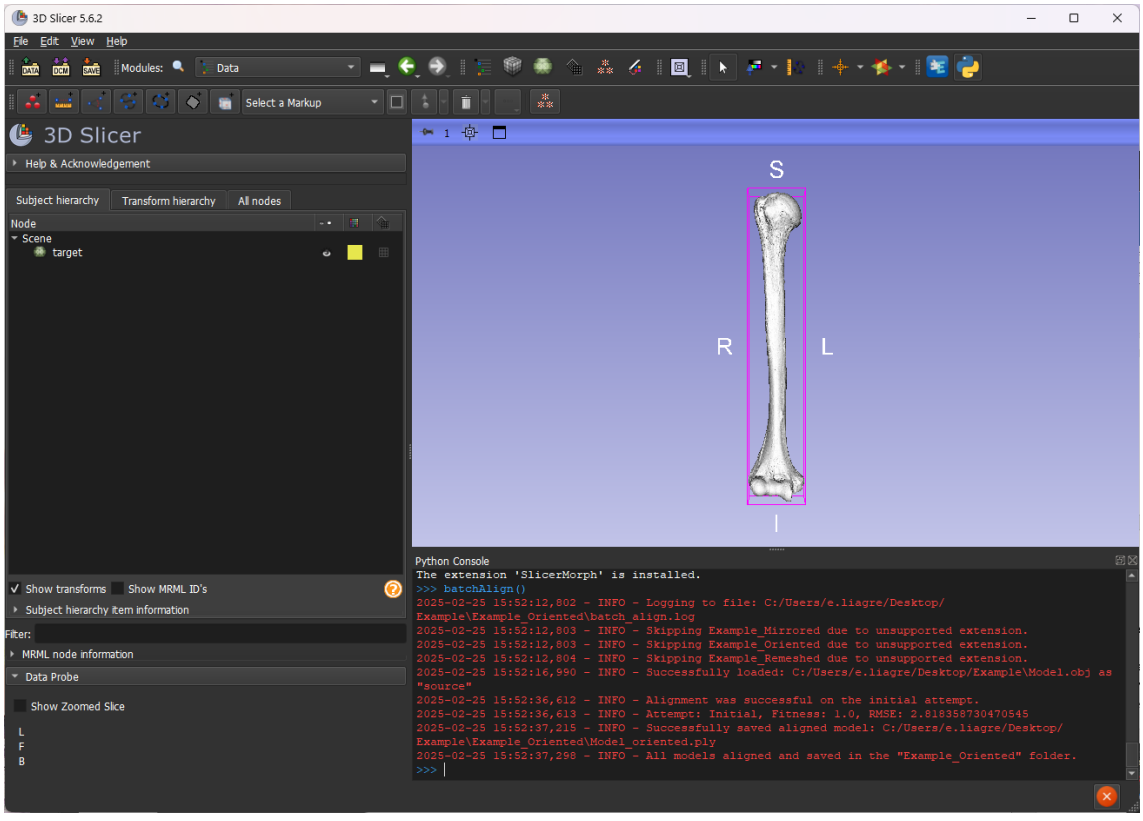


Select a folder (if no path is specified) – If you have not provided a folder path, a window will appear asking you to select the folder containing the models you want to align.

- Navigate to the desired folder.
- Click "Select Folder" (appears as "Sélectionner un dossier" in French in the screenshot).



Alignment Process – The script will create a new folder to store the aligned models, and will process all models in the selected folder. The process is complete when the Python Console displays the message: **"All models aligned and saved in the 'Example_Oriented' folder"** *(where "Example" is replaced by your folder's name)*, and when the console prompt (>>>) appears below the messages, indicating the script has finished running.



crop.py

Description

This function crops the humeral medial epicondyle as described in the associated article.

Pre-requisites

- **Required software:** 3D Slicer installed (See “How to prepare your Slicer environment, Step 1”)
- **Required script:** “crop” script added to 3D Slicer (See “How to prepare your Slicer environment, Step 2 to 4”)

Usage

To use the cropping function, run the following command in the Python Console:

```
crop(r'/path/to/your/file')
```

Arguments

`/path/to/your/file`	Path to the file. If omitted, a file dialog will prompt you to select a file. You can obtain the file path using the same method described in "How to Prepare Your Slicer Environment, Step 3, Method 2."

Details

This script uses the 3D Slicer environment to perform the steps outlined in the associated article in order to crop the humeral medial epicondyle. The process consists of the following steps: the placement of manual landmarks, generate automatic landmarks, verify landmark accuracy (the script will prompt the user to confirm or adjust the placement if needed), calculate the diameter of the circumscribed circle, generate the cropping plane and the curve, resample the curve, execute cropping using the defined plane and curve, extract the medial epicondyle, and export the final cropped model and landmark coordinates.

Note: This function is for right humeri only due to orientation requirements. If run for a left-sided model, the humeral diaphysis will be extracted instead of the medial epicondyle.

Example use of call

```
crop()
```

Runs the script and prompts for a file selection.

```
crop(r'/path/to/your/file)
```

Runs the script on a **specified file** (`/path/to/your/file`).

Manual Application

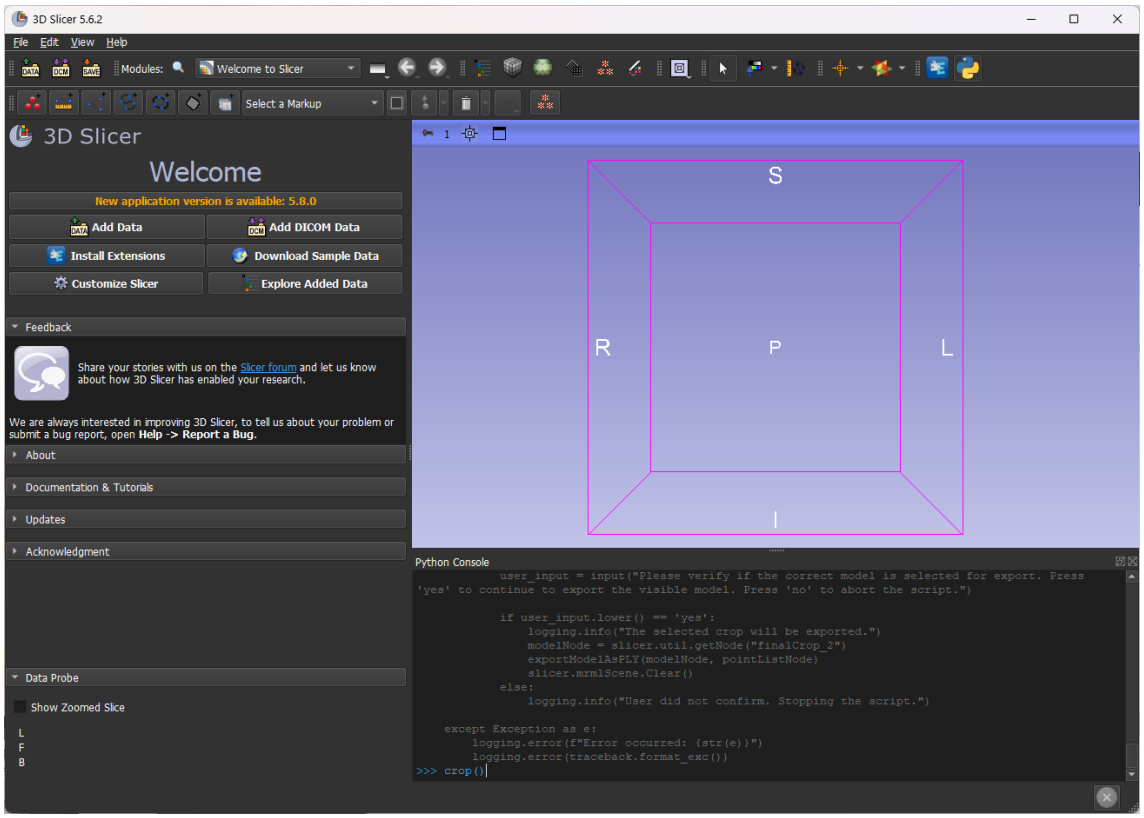
The procedure **cannot be fully executed manually** because some steps rely on automated calculations and scripting functions within 3D Slicer.

Visual Guide to Executing the Function

Call the function –In the Python Console, type:

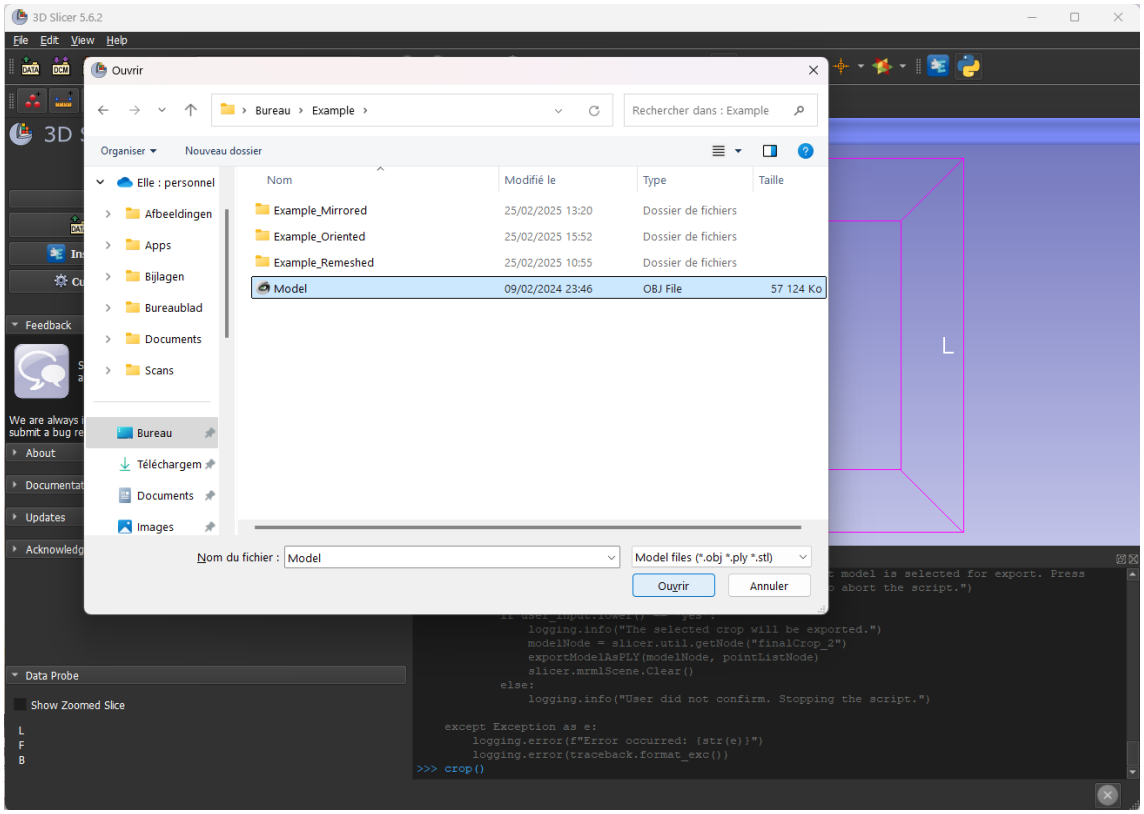
```
crop()
```

and press **Enter**.

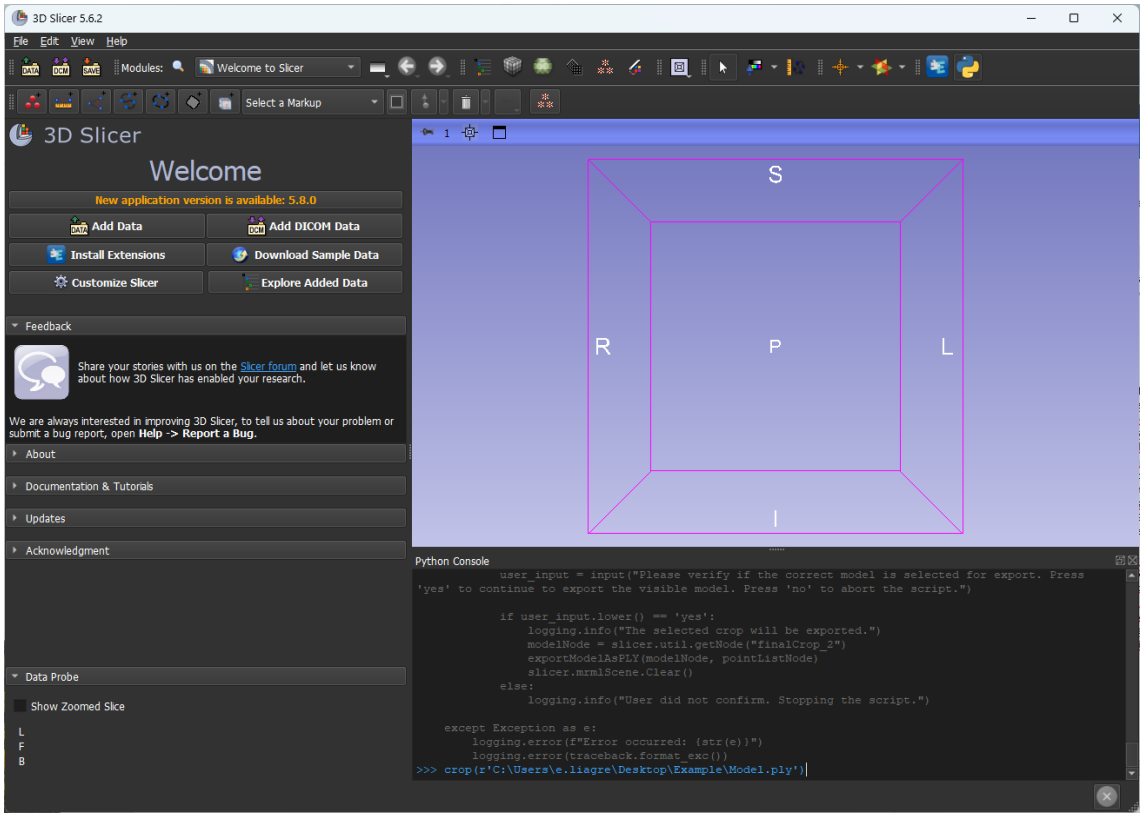


Select a file (if no path is specified) – If you have not provided a file path, a window will appear asking you to select the 3D model you want to crop.

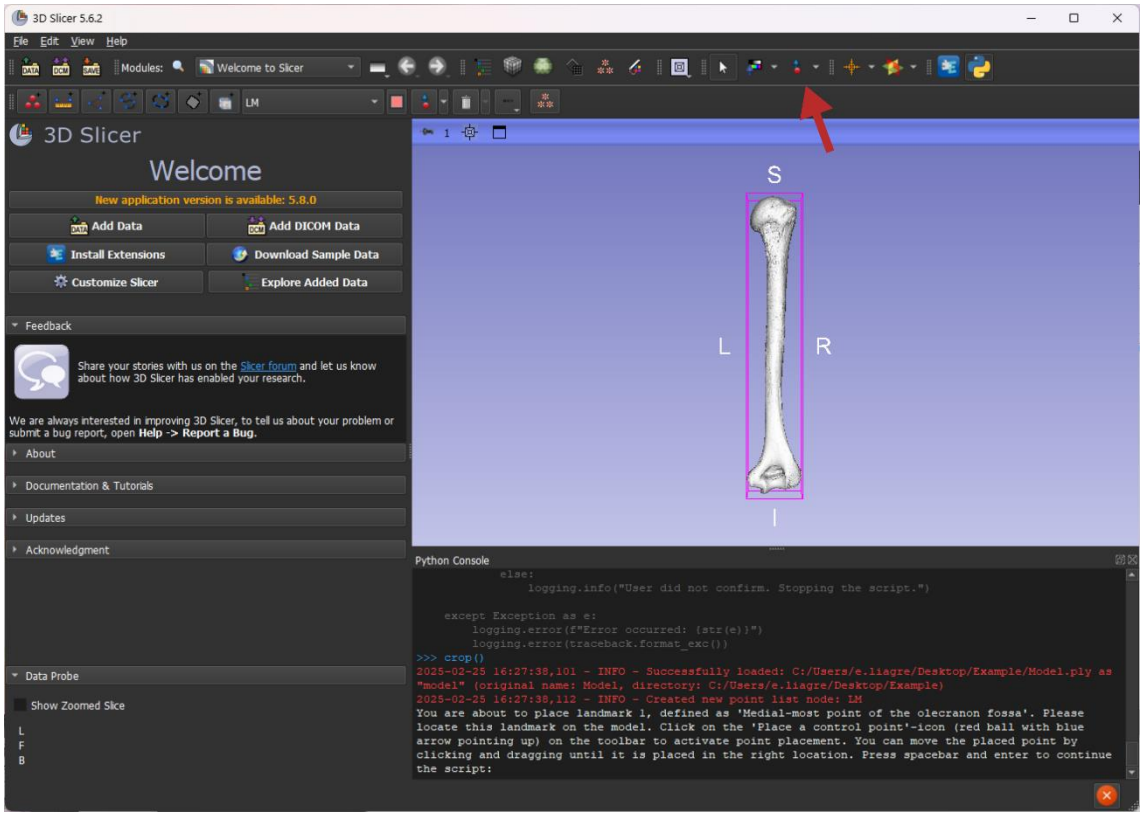
- Navigate to the desired file.
- Click "Open" (appears as "Ouvrir" in French in the screenshot).



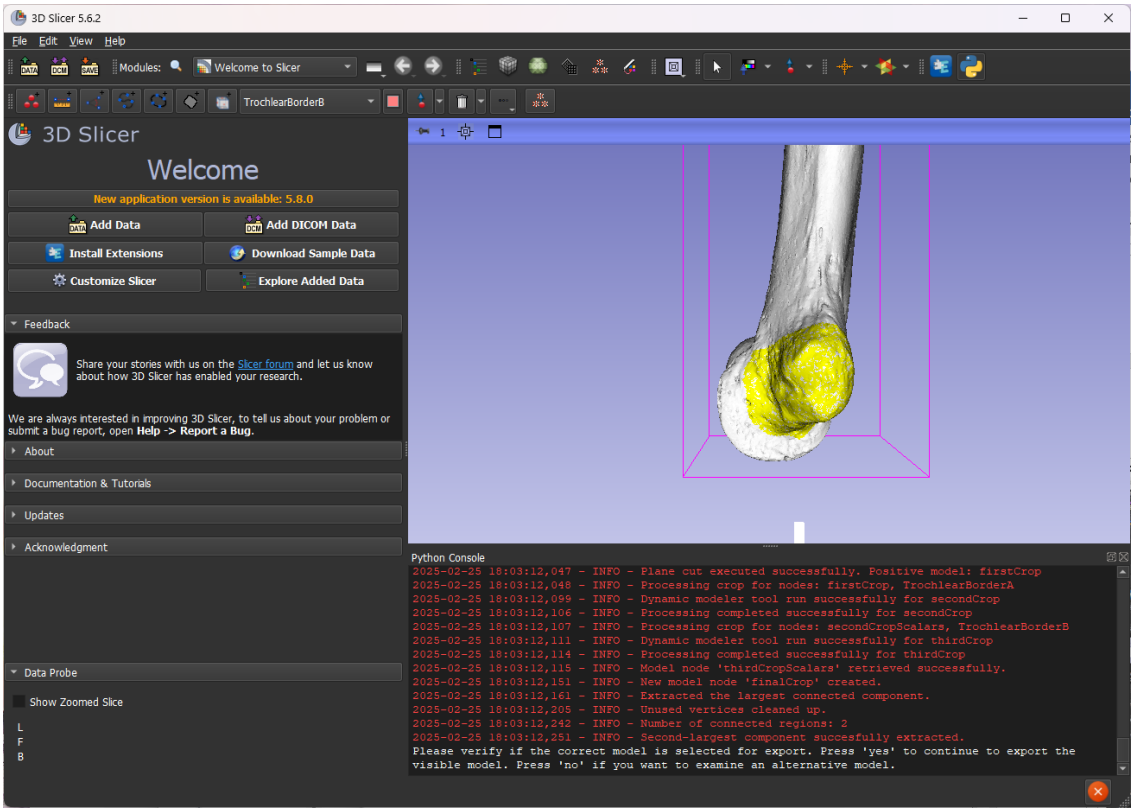
Alternatively, you can add the file path to function.



Cropping Process – Follow the instructions in the Python Console. The arrow in the screenshot indicates the “Place a control point”-icon.



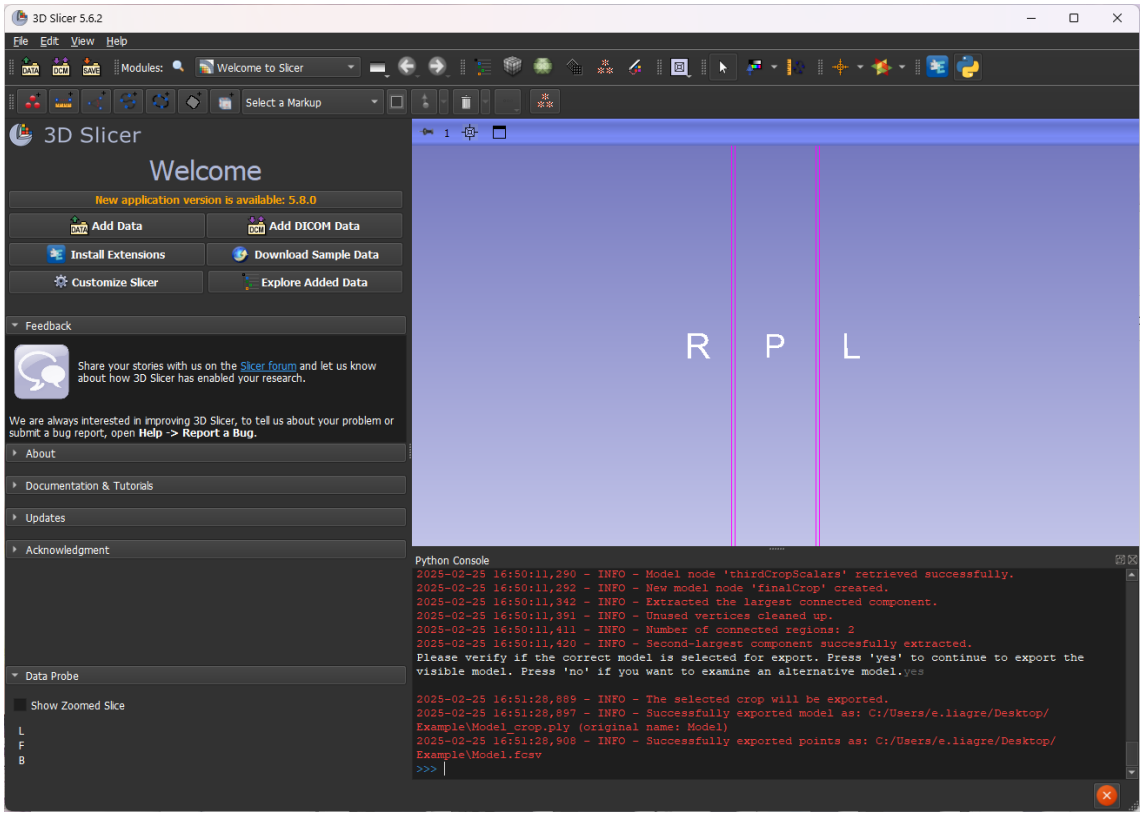
Export confirmation – Before exporting, the script will ask for confirmation.



The process is complete when the Python Console displays the message:

> **“Successfully exported model as: C:/Users/e.liagre/Desktop/Example\Model_crop.ply (original name: Model)”**
> **“Successfully exported points as: C:/Users/e.liagre/Desktop/Example\Model.fcsv”**

(where "Model" is replaced by your file’s name), and when the console prompt (>>>) appears below the messages, indicating the script has finished running.



batchCrop.py

Description

This batch function crops the humeral medial epicondyle, as described in the associated article, of all 3D models in a specified folder. If no folder path is provided, a file dialog will prompt you to select one. The cropped models are saved in a new folder within the original directory.

Pre-requisites

- **Required software:** 3D Slicer installed (See “How to prepare your Slicer environment, Step 1”)
- **Required script:** “batchCrop” script added to 3D Slicer (See “How to prepare your Slicer environment, Step 2 to 4”)

Usage

To use the batch cropping function, run the following command in the Python Console:

```
batchCrop(r'/path/to/your/folder')
```

Arguments

`/path/to/your/folder`	Path to the folder containing 3D models. If omitted, a file dialog will prompt you to select a folder. You can obtain the folder path using the same method described in "How to Prepare Your Slicer Environment, Step 3, Method 2."

Details

The script creates a new folder within the original directory to store the cropped models.

All models in the specified folder will be processed using the cropping method described in the associated article. The process consists of the following steps: the placement of manual landmarks, generate automatic landmarks, verify landmark accuracy (the script will prompt the user to confirm or adjust the placement if needed), calculate the diameter of the circumscribed circle, generate the cropping plane and the curve, resample the curve, execute cropping using the defined plane and curve, extract the medial epicondyle, and export the final cropped model and landmark coordinates.

Note: This function is for right humeri only due to orientation requirements. If run for a left-sided model, the humeral diaphysis will be extracted instead of the medial epicondyle.

Example use of call

```
batchCrop()
```

Runs the script and prompts for a folder selection.

```
batchCrop(r'/path/to/your/folder')
```

Runs the script on a **specified folder** (`/path/to/your/folder`).

Manual Application

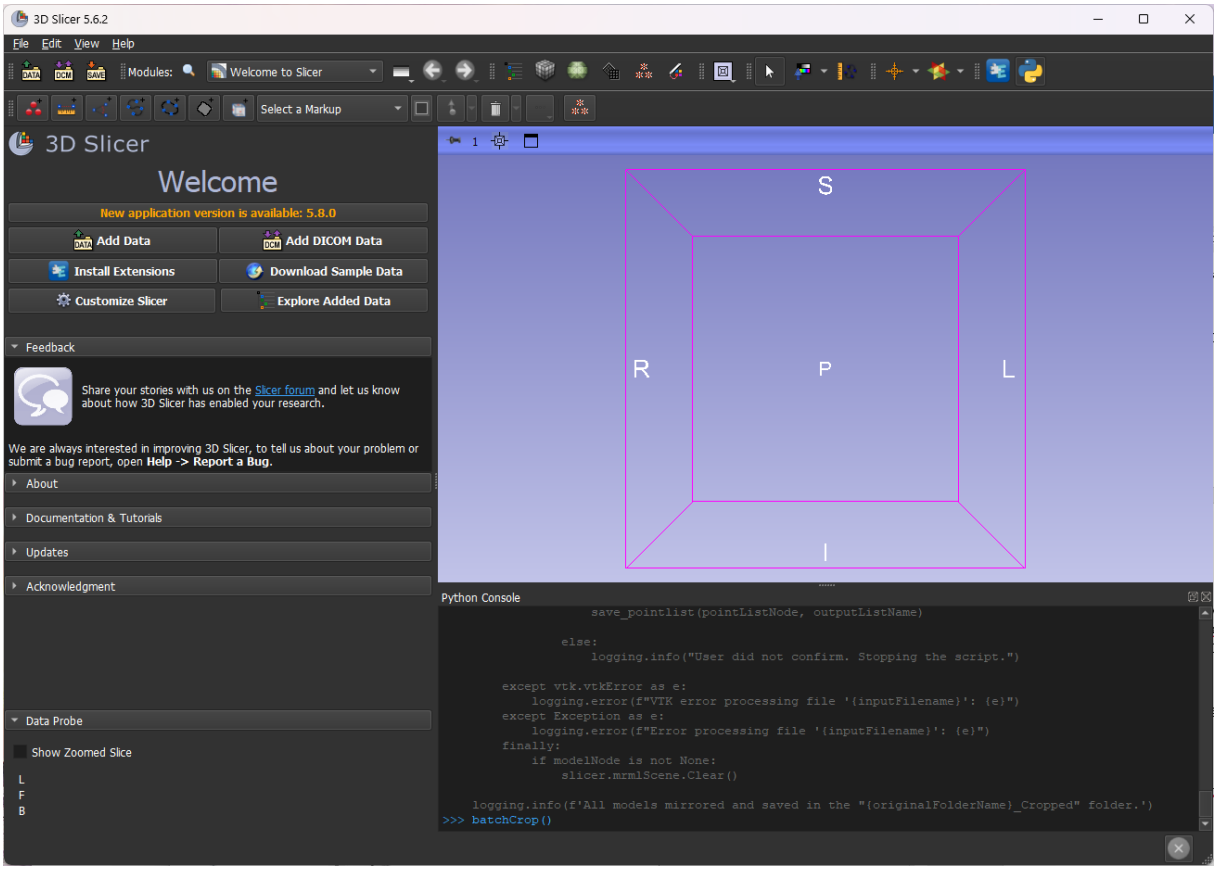
The “crop.py” script can be used to perform the cropping models individually. However, the procedure **cannot be fully executed manually** because some steps rely on automated calculations and scripting functions within 3D Slicer.

Visual Guide to Executing the Function

Call the function – In the Python Console, type:

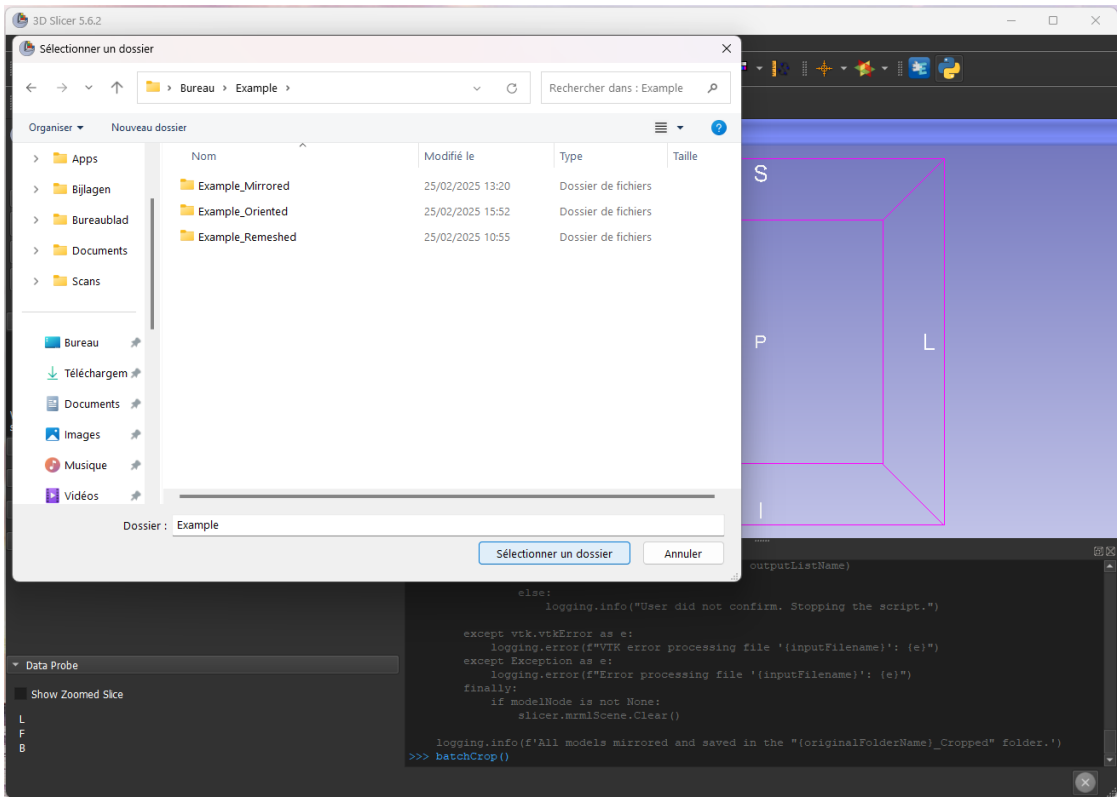
```
batchCrop()
```

and press **Enter**.

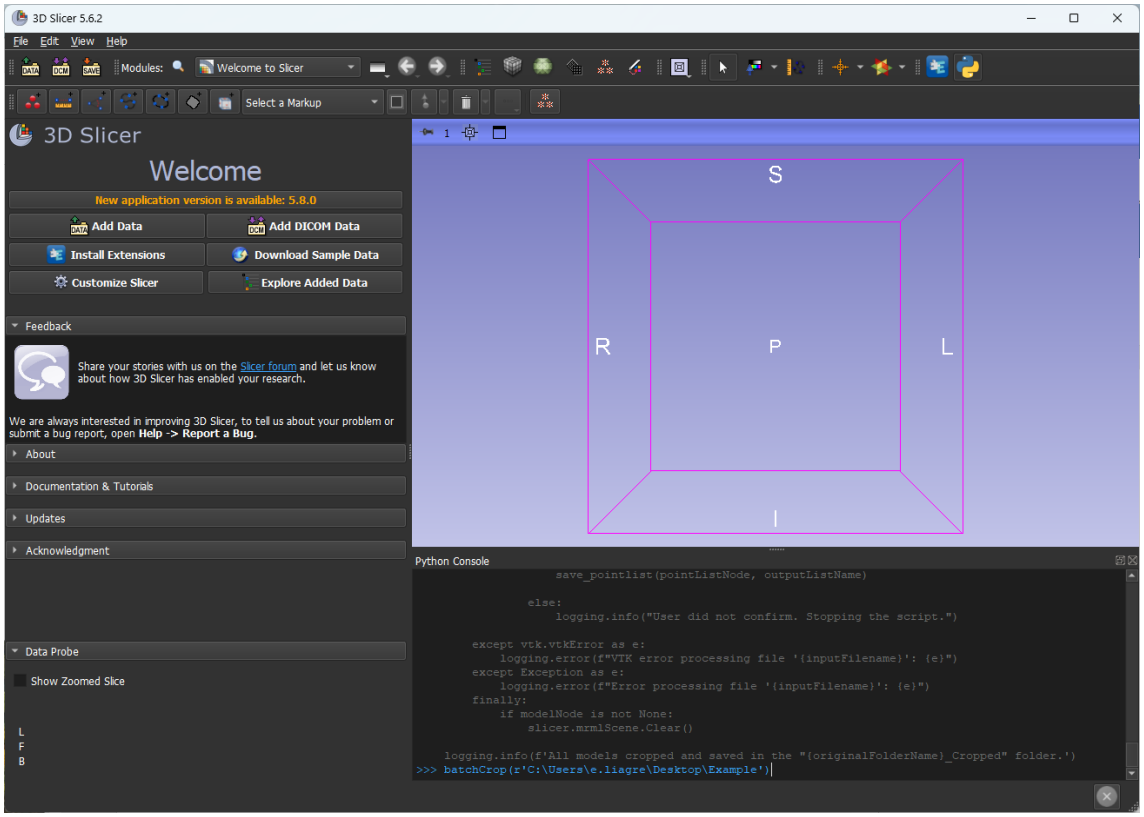


Select a folder (if no path is specified) – If you have not provided a folder path, a window will appear asking you to select the folder containing the models you want to crop.

- Navigate to the desired folder.
- Click "Select Folder" (appears as "Sélectionner un dossier" in French in the screenshot).



Alternatively, you can add the file path to function.



Cropping Process – The script will create a new folder to store the cropped models, and will process all models in the selected folder. Follow the instructions in the Python Console (refer to `crop.py` for details). The process is complete when the Python Console displays the message: **"All models cropped and saved in the 'Example_Cropped' folder"** *(where "Example" is replaced by your folder's name)*, and when the console prompt (>>>) appears below the messages, indicating the script has finished running.