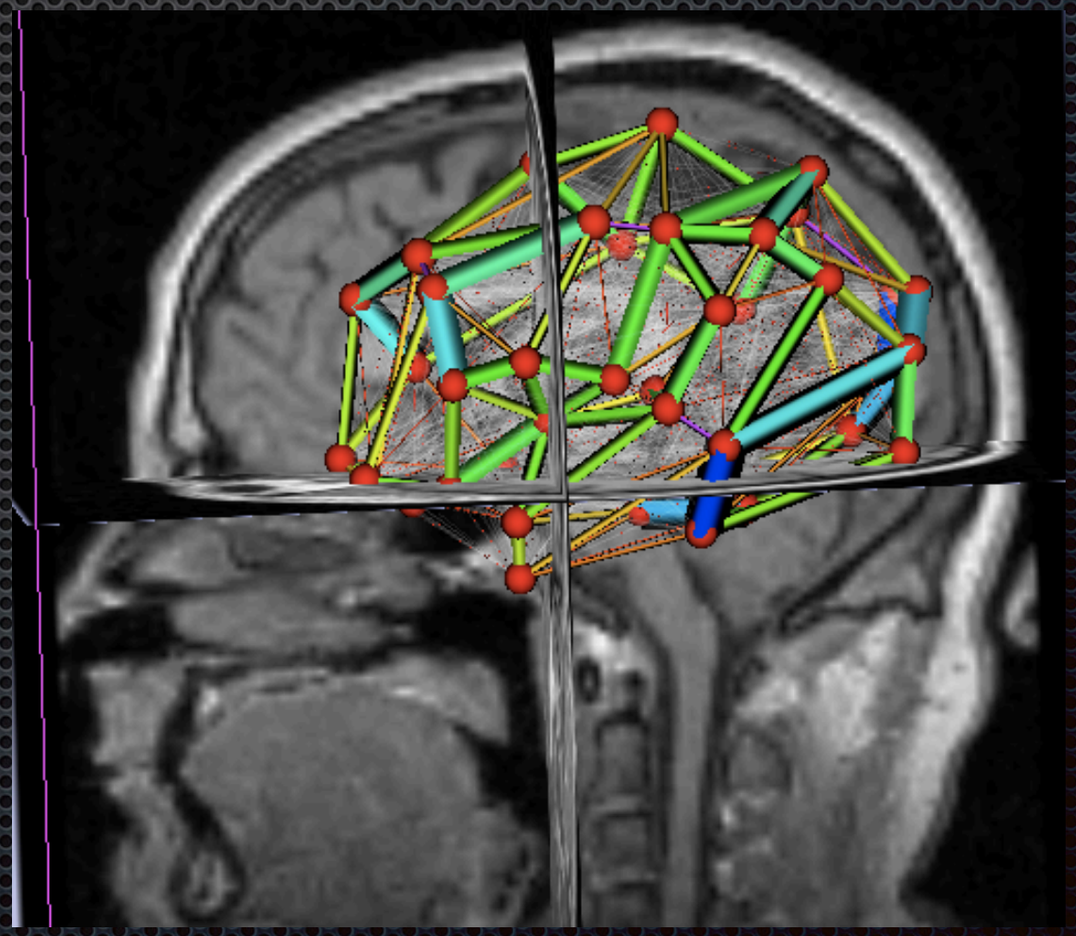
Brain Connectome Visualization

Developper handbook

Contributors: Wieke Prummel (CPE),
Dr Juan Prieto (NIRAL, UNC)
Dr Martin Styner (NIRAL, UNC)



Code structure: visuThreeD.py

- The code is divided into 3 classes:

1) Widgets class (where we define all the user interface widgets)

2) Logic class (where we define the functions linked to the user interface, in order to make this interface active)

3) Test class (where we test the existing functions with default values)

1. Widgets class

```python
class visuThreeDWidget(ScriptedLoadableModuleWidget):
  """Uses ScriptedLoadableModuleWidget base class, available at:
  https://github.com/Slicer/Slicer/blob/master/Base/Python/slicer/ScriptedLoadableModule.py
  """

  def setup(self):
    ScriptedLoadableModuleWidget.setup(self)

    ####################################
    # Import node graph json file Area
    ####################################
    self.fileCollapsibleButton = ctk.ctkCollapsibleButton()
    self.fileCollapsibleButton.text = "Import Node Graph Json File"
    self.layout.addWidget(self.fileCollapsibleButton)
    self.fileImportFormLayout = qt.QFormLayout(self.fileCollapsibleButton)

    self.fileImport = ctk.ctkPathLineEdit()
    self.fileImport.filters = ctk.ctkPathLineEdit.Files
    self.fileImport.settingKey = 'JsonInputFile'
    self.fileImportFormLayout.addRow("Input Json File:", self.fileImport)

    self.fileImportButton = qt.QPushButton('Load File')
    self.fileImportButton.checked = False
    self.fileImportFormLayout.addRow(self.fileImportButton)
```

This type of comment indicates that the widgets that follow are for example found in the section where you import the Json File

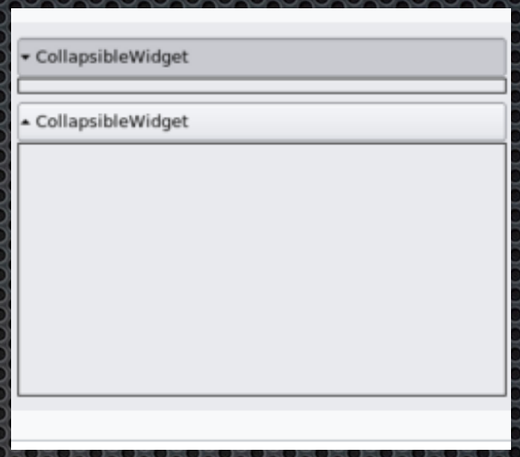
The order in which the widgets are created is the order in which they will appear in the user interface

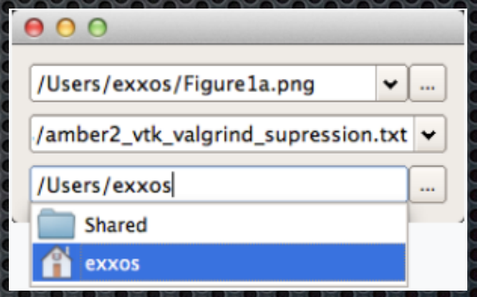
```python
def setup(self):
  ScriptedLoadableModuleWidget.setup(self)

  ##################################
  # Import node graph json file Area
  ##################################
  self.fileCollapsibleButton = ctk.ctkCollapsibleButton()
  self.fileCollapsibleButton.text = "Import Node Graph Json File"
  self.layout.addWidget(self.fileCollapsibleButton)
  self.fileImportFormLayout = qt.QFormLayout(self.fileCollapsibleButton)

  self.fileImport = ctk.ctkPathLineEdit()
  self.fileImport.filters = ctk.ctkPathLineEdit.Files
  self.fileImport.settingKey = 'JsonInputFile'
  self.fileImportFormLayout.addRow("Input Json File:", self.fileImport)

  self.fileImportButton = qt.QPushButton('Load File')
  self.fileImportButton.checked = False
  self.fileImportFormLayout.addRow(self.fileImportButton)
```





Load File

Documentation on Qt widgets is found at : https://doc.qt.io/qt-5/

Documentation on ctk widgets is found at : https://commontk.org/index.php/Documentation/ImageGallery

```
##################################
# Connections
##################################
self.coord = []
self.index = []
self.position = []
self.visu = []
self.fileImportButton.connect('clicked(bool)', self.on_node_graph_json_load)
self.inputSelector.connect("nodeActivated(vtkMRMLNode*)", self.on_select)
self.regionButtons.connect('checkedIndexesChanged()', self.on_regions_checked)
self.calculateAllFilteredregionsButton.connect('clicked(bool)', self.on_select_filtered_regionButtons)
self.calculateAllregionsButton.connect('clicked(bool)', self.on_select_all_regionButtons)
self.calculateNoregionsButton.connect('clicked(bool)', self.on_deselect_all_regionButtons)
self.calculateNoFilteredregionsButton.connect('clicked(bool)', self.on_deselect_filtered_regionButtons)
self.regionSearchBox.connect("textChanged(QString)", self.on_search)
self.ColorTable.connect("currentNodeChanged(vtkMRMLNode*)", self.on_node_color_clicked)
self.nodeThresholdSliderWidget.connect("valuesChanged(double, double)", self.sliderbar_changed)
self.nodeMinSizeSpinBox.connect("valueChanged(double)", self.min_nodesize_changed)
self.nodeMaxSizeSpinBox.connect("valueChanged(double)", self.max_nodesize_changed)
self.tableStartSpinBox.connect("valueChanged(double)", self.table_start_changed)
self.matrixConnectSelector.connect("nodeActivated(vtkMRMLNode*)",self.on_select_matrix)
self.connectionThresholdSliderWidget.connect("valuesChanged(double, double)", self.sliderbar2_changed)
self.maxConnectionSpinBox.connect("valueChanged(double)", self.max_connection_changed)
self.connectionColorTable.connect("currentNodeChanged(vtkMRMLNode*)", self.on_connect_color_clicked)
self.fscaleDoubleSpinBox.connect("valueChanged(double)", self.on_fscale_changed)
self.logScaleDoubleSpinBox.connect("valueChanged(double)", self.on_logscale_changed)
```

Section where we connect the widgets

1st argument is the signal that activates the widget

2nd argument is where you call the function linked to the widget

```
self.ColorTable.connect("currentNodeChanged(vtkMRMLNode*)", self.on_node_color_clicked)
```

```
# When Color map for Nodes is loaded this function is called
def on_node_color_clicked(self, color_map):
  self.logic.set_node_color_map(color_map)
  self.logic.update()
```

The functions in the widgets class are called in the second argument of the widget's connect function.
"On_node_color_clicked" is activated when the user selects a colour map in the drop down menu.
The id of the colour map is then given to the logic part "self.logic.function()" in order to attribute this colour to the objects.

2. Logic Class

```
def __init__(self):
  self.coord = []
  self.resNameRegion = []
  self.segmentationNode = slicer.vtkMRMLModelDisplayNode()
  self.index = []
  self.position = []
  self.visu = []
```

The init function, is the initialise function

Then we define all the other functions such as the set_node_graph_json()

```
def set_node_graph_json(self, path_json):
  if os.path.isfile(path_json):
      self.path_to_json = path_json
  else:
      raise Exception("File does not exist")
```

How json_extract_properties.py is linked to visuThreeD.py

json_extract_properties is a separate file we create to read our Json and csv files.
To access this code:
"import json_extract_properties as JEP"

We define self.jep in the logic class, init function:

```
self.path_to_json = ''
self.jep = JEP.json_extract_properties()
```

This makes it easier to call a function from json_extract_properties() in a function from the logic class in visuThreeD.py

```
def set_user_file(self, user_file):
    self.user_file = user_file
    self.jep = JEP.json_extract_properties()
    self.jep.set_csv_file(self.user_file)
    self.jep.read_csv()
```

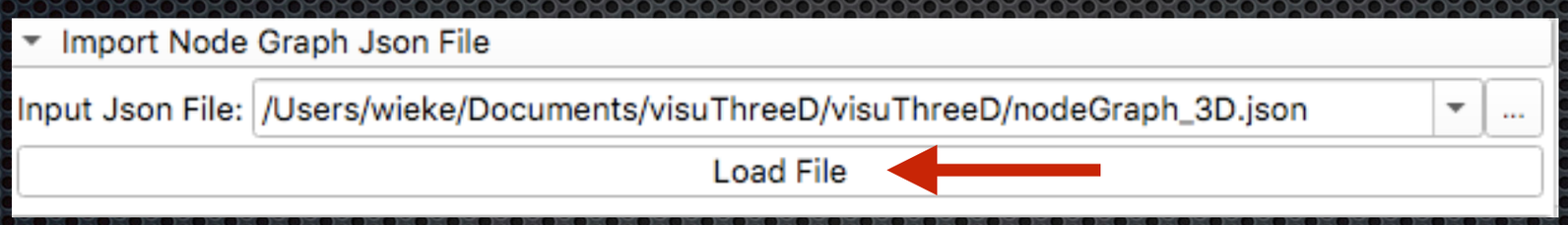
Node Graph Json File

- User input : **Json file**
- This node information is seen as a dictionary where each key associates a value
- **'Coord'** info allows us to initialise the center of our spheres in the renderer
- We create a list of coordinates indexed by MatrixRow


```
[
  {
    "VisuOrder": 78,
    "MatrixRow": 1,
    "name": "Precentral_L",
    "VisuHierarchy": "seed.left.frontal.",
    "coord": [
      -38.649999999999999,
      -5.6799999999999997,
      50.939999999999998
    ],
    "labelValue": "1",
    "AAL_ID": 1
  },
  {
    "VisuOrder": 1,
    "MatrixRow": 2,
    "name": "Precentral_R",
    "VisuHierarchy": "seed.right.frontal.",
    "coord": [
      41.369999999999997,
      -8.2100000000000009,
      52.090000000000003
    ],
    "labelValue": "2",
    "AAL_ID": 2
  },
```

3. Limits of our code and what could be improved

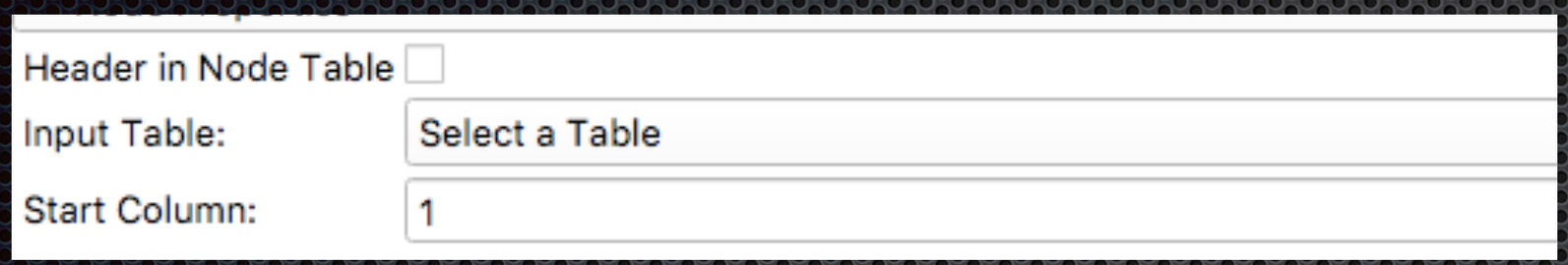
Currently the user loads the Json file and then has to click on the **Load File button** in order to access the set_node_graph_json() function in the logic that reads the son file.



Suggestion: make this more intuitive for the user.
For example with:
- a pop up window
- or forget the button and access the Json file right after it's loaded

Node Table section:

Once the table Node is loaded through the Data Module of Slicer, even if the name of the table appears in the “select a Table” box, the user still has to click on this table in order for it to be selected.



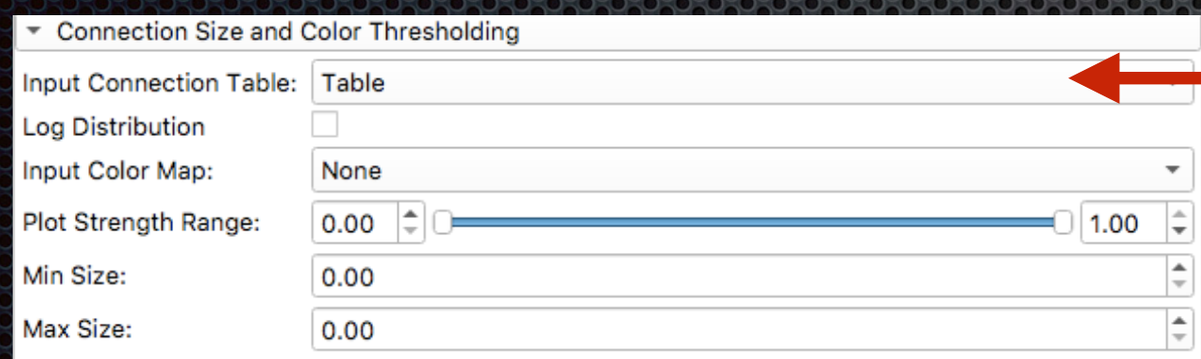
This is because of the signal that is used to activate this widget. This should be changed for this module to become more fluid

```
self.inputSelector = slicer.qMRMLNodeComboBox()
self.inputSelector.nodeTypes = ["vtkMRMLTableNode"]
self.inputSelector.selectNodeUponCreation = True
self.inputSelector.addEnabled = False
self.inputSelector.removeEnabled = False
self.inputSelector.noneEnabled = False
self.inputSelector.showHidden = False
self.inputSelector.showChildNodeTypes = False
self.inputSelector.setMRMLScene( slicer.mrmlScene )
self.inputSelector.setToolTip( "The input file loaded trough the import Data module should be a one line table
parametersFormLayout.addRow("Input Table: ", self.inputSelector)
```

self.inputSelector.connect("nodeActivated(vtkMRMLNode*)", self.on_select)

Connection Matrix section:

See with Kitware (Slicer discourse page) how we could import a 78x78 (AAL) or a Destrieux matrix into the existing Slicer Table module without having to ask the user to define a header in their csv file before loading it through the Data module.



Connection Matrix Input

Currently we ask the user to add a header of strings into their csv file

	A	B	C	D	E	F	G	H
1	PreCG.L	PreCG.R	SFGdor.L	SFGdor.R	ORBsup.L	ORBsup.R	MFG.L	MFG.R
2	0	2.30481138373834E-05	0.002370561773804	5.80856714174019E-06	1.71046519640654E-06	2.45508411471601E-07	0.003707346980583	4.02256089565
3	2.30481138373834E-05	0	1.1479541657271E-05	0.002790105972168	8.09368389466817E-08	1.6457157252492E-06	4.69433665890754E-07	0.00343223457
4	0.002370561773804	1.1479541657271E-05	0	0.000704622630397	0.001183631124335	0.000203434744692	0.007878157186237	2.33340906683
5	5.80856714174019E-06	0.002790105972168	0.000704622630397	0	7.69439548919788E-06	0.001114783551232	6.82918068085785E-05	0.00810426233
6	1.71046519640654E-06	8.09368389466817E-08	0.001183631124335	7.69439548919788E-06	0	0.00013942179877	0.000140007241905	6.74473657889
7	2.45508411471601E-07	1.6457157252492E-06	0.000203434744692	0.001114783551232	0.00013942179877	0	2.61695779260938E-07	0.000110303422
8	0.003707346980583	4.69433665890754E-07	0.007878157186237	6.82918068085785E-05	0.000140007241905	2.61695779260938E-07	0	6.53430079762
9	4.02256089565008E-06	0.003432234571319	2.33340906683283E-05	0.008104262339519	6.74473657889014E-08	0.000110303422011	6.53430079762877E-06	0
10	2.69789463155606E-08	0	0.000173372104813	1.6457157252492E-07	0.002968345078896	3.31841039681395E-07	0.00160931573088	2.69789463155
11	2.69789463155606E-09	8.36347335782378E-08	9.22679963992172E-07	0.000221348765046	1.61873677893363E-07	0.0033870151594	5.39578926311212E-09	0.002282348713
12	0.001692076346598	1.0683662740962E-06	0.000228487394241	1.61064309503897E-06	1.10613679893798E-07	5.39578926311212E-09	0.000952443137567	5.66557872626
13	1.64031993598608E-06	0.002374411669443	2.35256411871688E-06	4.69055960642336E-05	8.09368389466817E-09	2.37414727576933E-07	5.23391558521875E-07	0.001101666387
14	0.000174132911099	2.64393673892494E-07	5.30325147724974E-05	1.78573645662695E-05	3.66913669891624E-07	3.23747355786727E-08	0.004220089948994	3.80942721975
15	4.91016822943202E-07	2.61102242441995E-05	3.61517880628512E-06	5.21125327031368E-05	5.39578926311212E-09	2.58997884629382E-07	1.15469890230599E-06	0.004225129616
16	5.7465155652144E-07	8.09368389466817E-09	1.77737298326913E-05	1.48384204735583E-07	0.002918698421887	7.43269970993694E-06	0.000219274084074	2.69789463155
17	2.69789463155606E-09	4.31663141048969E-08	2.75185252418718E-07	7.31938813541158E-05	2.98656935713256E-06	0.003426852271529	4.5864208736453E-08	0.00025399329
18	0.000370447911859	2.07737886629816E-07	3.41688355086575E-05	1.75363151051144E-07	3.58819985996956E-07	0	4.00097773859763E-06	5.12599979995
19	9.1728417472906E-08	0.000788993889209	5.12599979995651E-08	1.96946308103592E-07	0	6.20515765257893E-08	2.69789463155606E-09	5.36881031679
20	2.90806062335427E-05	4.77257560322267E-06	0.002068527074012	0.000249541763946	1.0251999599913E-06	2.99466304102722E-07	2.32720390918025E-05	6.10263765657
21	5.60622504437349E-06	3.47542786437051E-05	0.000222387454479	0.004077689674551	3.64215775260068E-07	4.91016822943202E-07	1.48114415272428E-05	4.70944486884