

# *Implement a new Subject*

*The BRAPH 2 Developers*

*September 13, 2023*

This is the developer tutorial for implementing a new Subject. In this Tutorial, we will explain how to create the generator file \*.gen.m for a new subject, which can then be compiled by braph2genesis. All types of subjects are extensions of the base element Subject. Here, we will use as examples the subjects SubjectCon (subject with connectivity data), SubjectCON\_MP (subject with connectivity multiplex data), SubjectFUN (subject with functional data), SubjectFUN\_MP (subject with functional multiplex data), SubjectST (subject with structural data), and SubjectST\_MP (subject with structural multiplex data).

## *Contents*

<i>Implementation of a subject with connectivity matrix</i>	<i>2</i>
<i>subject with connectivity data (SubjectCON)</i>	<i>2</i>
<i>subject with connectivity multiplex data (SubjectCON_MP)</i>	<i>5</i>
<i>Implementation of a subject with functional data</i>	<i>8</i>
<i>subject with functional data (SubjectFUN)</i>	<i>8</i>
<i>subject with functional multiplex data (SubjectFUN_MP)</i>	<i>11</i>
<i>Implementation of a subject with structural data</i>	<i>14</i>
<i>subject with structural data (SubjectST)</i>	<i>14</i>
<i>subject with structural multiplex data (SubjectST_MP)</i>	<i>17</i>

## Implementation of a subject with connectivity matrix

### subject with connectivity data (SubjectCON)

We will start by implementing in detail SubjectCON. The connectivity matrix can be obtained from DTI data.

**Code 1: SubjectCON element header.** The header section of the generator code for `_SubjectCON.gen.m` provides the general information about the SubjectCON element.

---

```

1 %% iheader!
2 SubjectCON < Subject (sub, subject with connectivity matrix) is a subject
   with connectivity matrix (e.g. DTI). ①
3
4 %% idescription!
5 Subject with a connectivity matrix (e.g. obtained from DTI).
6
7 %% iseealso! ②
8 ImporterGroupSubjectFUN_TXT, ExporterGroupSubjectFUN_TXT,
   ImporterGroupSubjectFUN_XLS, ExporterGroupSubjectFUN_XLS

```

---

① The element SubjectCON is defined as a subclass of Subject. The moniker will be sub.

② allows menu to import and export text and Excel spreadsheet files.

**Code 2: SubjectCON element prop update.** The `props_update` section of the generator code for `_SubjectCON.gen.m` updates the properties of the Subject element. This defines the core properties of the subject.

---

```

1 %% iprops_update!
2
3 %% iprop!
4 NAME (constant, string) is the name of the subject.
5 %%%% idefault!
6 'SubjectCON'
7
8 %% iprop!
9 DESCRIPTION (constant, string) is the description of the subject.
10 %%%% idefault!
11 'SubjectCON with a connectivity matrix (e.g. obtained from DTI).'
12
13 %% iprop!
14 TEMPLATE (parameter, item) is the template of the subject.
15 %%%% isettings!
16 'SubjectCON'
17
18 %% iprop!
19 ID (data, string) is a few-letter code for the subject.
20 %%%% idefault!
21 'SubjectCON ID'
22
23 %% iprop!
24 LABEL (metadata, string) is an extended label of the subject.
25 %%%% idefault!
26 'SubjectCON label'
27
28 %% iprop!
29 NOTES (metadata, string) are some specific notes about the subject.
30 %%%% idefault!

```

```

31 'SubjectCON notes'
32
33 %% iprops!
34
35 %% iprop!
36 BA (data, item) is a brain atlas.
37 %%%% isettings!
38 'BrainAtlas'
39
40
41 %% iprop!
42 CON (data, smatrix) is an adjacency matrix.
43 %%%% icheck_value!
44 br_number = sub.get('BA').get('BR_DICT').get('LENGTH'); ①
45 check = isequal(size(value), [br_number, br_number]); ②
46 if check ③
47     msg = 'All ok!';
48 else
49     msg = ['CON must be a square matrix with the dimension equal to the
50           number of brain regions (' int2str(br_number) ').'];
51 end
52 %%%% igui! ④
53 pr = PanelPropMatrix('EL', sub, 'PROP', SubjectCON.CON, ...
54     'ROWNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
55     'COLUMNNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
56     varargin{:});

```

---

① defines the number of brain regions from the Brain Atlas.

② The value is the data matrix. Checking the size of value is equal to the number of brain regions.

③ returns the check information msg according to the variable check.

④ plots the panel of a property matrix-like with element sub and the property number SubjectCon.Con. ROWNAME and COLUMNNAME are the name of regions from brain atlas.

Code 3: **SubjectCON element tests.** The tests section from the element generator `_SubjectCON.gen.m`. A general test should be prepared to test the properties of the Subject when it is empty and full. Furthermore, additional tests should be prepared for the rules defined.

```

1 %% itests!
2
3 %%% itest!
4 %%% iname!
5 GUI ①
6 %%% iprobability! ②
7 .01
8 %%% icode!
9 im_ba = ImporterBrainAtlasXLS('FILE', 'desikan_atlas.xlsx'); ③
10 ba = im_ba.get('BA'); ④
11
12 gr = Group('SUB_CLASS', 'SubjectCON', 'SUB_DICT', IndexedDictionary('
    IT_CLASS', 'SubjectCON')); ⑤
13 for i = 1:1:50 ⑥
14     sub = SubjectCON( ... ⑦
15         'ID', ['SUB CON ' int2str(i)], ...
16         'LABEL', ['Subejct CON ' int2str(i)], ...
17         'NOTES', ['Notes on subject CON ' int2str(i)], ...
18         'BA', ba, ...
19         'CON', rand(ba.get('BR_DICT').get('LENGTH')) ...
20     );
21     sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
        rand())) ⑧
22     sub.memorize('VOI_DICT').get('ADD', VOICategoric('ID', 'Sex', '
        CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1))) ⑨
23     gr.get('SUB_DICT').get('ADD', sub) ⑩
24 end
25
26 gui = GUIElement('PE', gr, 'CLOSEREQ', false); ⑪
27 gui.get('DRAW') ⑫
28 gui.get('SHOW') ⑬
29
30 gui.get('CLOSE') ⑭

```

① checks that GUI is constructing well.

② assigns a low test execution probability.

③ imports the brain atlas desikan from the file 'desikan\_atlas.xlsx'. There are also other atlases in Graph2 folder atlases, including aal90\_atlas.xlsx, aal116\_atlas.xlsx, bna\_atlas.xlsx, craddock\_atlas.xlsx, desikan\_subcortical\_atlas.xlsx, destrieux\_atlas.xlsx, destrieux\_subcortical\_atlas.xlsx, schaefer200\_atlas.xlsx and subcortical\_atlas.xlsx.

④ returns the brain atlas.

⑤ represents a group of subjects whose class is defined in the property 'SUB\_CLASS'. 'SUB\_DICT' manages the subjects as an indexed dictionary of subjects.

⑥ construts 50 subjects.

⑦ defines the 'ID', 'LABEL', 'NOTES', 'BA' (Brain Atlas) and 'CON' (a random adjacency matrix) for a subject.

⑧ adds a random Numeric 'Age' as the variable of interest of the subject.

⑨ adds a random Categoric 'Sex' as the variable of interest of the subject.

⑩ adds 'sub' into group.

⑪ constructs the GUI panel from gr. Setting the 'CLOSEREQ' to false means doesn't confirm whether the GUI is close.

⑫ draws the contents of a GUI before showing it.

⑬ shows the figure and its dependent figures.

⑭ closes the figure and its dependent figures.

### *subject with connectivity multiplex data (SubjectCON\_MP)*

We can now use SubjectCON as the basis to implement the SubjectCON\_MP. The parts of the code that are modified are highlighted. The multi-layer data allows connections between any nodes across the multiple layers. The SubjectCON\_MP can also be used on ordinal multilayer data.

**Code 4: SubjectCON\_MP element header.** The header section of the generator code for `_SubjectCON_MP.gen.m` provides the general information about the SubjectCON\_MP element. [← Code 1](#)

---

```

1
2 %% iheader!
3 SubjectCON_MP < Subject (sub, subject with connectivity multiplex data) is a
  subject with connectivity multiplex data.
4
5 %%% idescription!
6 Subject with L connectivity matrices (e.g. obtained from DTI).
7
8 %%% iseealso!
9 ImporterGroupSubjectCON_MP_TXT, ExporterGroupSubjectCON_MP_TXT,
  ImporterGroupSubjectCON_MP_XLS, ExporterGroupSubjectCON_MP_XLS

```

---

**Code 5: SubjectCON\_MP element prop update.** The `props_update` section of the generator code for `_SubjectCON_MP.gen.m` updates the properties of the Subject element. [← Code 2](#)

---

```

1 %% iprops_update!
2
3 %%% iprop!
4 NAME (constant, string) is the name of the subject.
5 %%% idefault!
6 'SubjectCON_MP'
7
8 %%% iprop!
9 DESCRIPTION (constant, string) is the description of the subject.
10 %%% idefault!
11 'Subject with L connectivity matrices (e.g. obtained from DTI).'
12
13 %%% iprop!
14 TEMPLATE (parameter, item) is the template of the subject.
15 %%% isettings!
16 'SubjectCON_MP'
17
18 %%% iprop!
19 ID (data, string) is a few-letter code for the subject.
20 %%% idefault!
21 'SubjectCON_MP ID'
22
23 %%% iprop!
24 LABEL (metadata, string) is an extended label of the subject.
25 %%% idefault!
26 'SubjectCON_MP label'
27
28 %%% iprop!
29 NOTES (metadata, string) are some specific notes about the subject.
30 %%% idefault!

```

---

```

31 'SubjectCON_MP notes'
32
33 %% iprops!
34
35 %%% iprop!
36 BA (data, item) is a brain atlas.
37 %%% isettings!
38 'BrainAtlas'
39
40 %%% iprop!
41 L (data, scalar) is the number of layers of subject data. ①
42 %%% ndefault!
43 2 ②
44
45 %%% iprop!
46 LAYERLABELS (metadata, stringlist) are the layer labels provided by the user
   . ③
47
48 %%% iprop!
49 ALAYERLABELS (query, stringlist) returns the processed layer labels. ④
50 %%% icalculate!
51 value = sub.get('LAYERLABELS'); ⑤
52
53 %%% iprop!
54 CON_MP (data, cell) is a cell containing L matrices corresponding
   connectivity matrices of each layer.
55 %%% ickheck_value!
56 br_number = sub.get('BA').get('BR_DICT').get('LENGTH');
57 num_layers = sub.get('L'); ⑥
58 check = (iscell(value) && isequal(length(value), num_layers) && isequal(
   cellfun(@(v) size(v, 1), value), ones(1, num_layers) * br_number) &&
   isequal( cellfun(@(v) size(v, 2), value), ones(1, num_layers) *
   br_number)) || (isempty(value) && br_number == 0); ⑥
59 if check
60     msg = 'All ok!';
61 else
62     msg = ['CON_MP must be a cell with L square matrices with the dimension
   equal to the number of brain regions (' int2str(br_number) ').'];
63 end
64 %%% igui!
65 pr = PanelPropCell('EL', sub, 'PROP', SubjectCON_MP.CON_MP, ...
66     'TABLE_HEIGHT', s(40), ... ⑦
67     'XSLIDERSHOW', true, ... ⑧
68     'XSLIDERLABELS', sub.getCallback('ALAYERLABELS'), ... ⑨
69     'YSLIDERSHOW', false, ... ⑩
70     'ROWNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
71     'COLUMNNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
72     varargin{:});

```

① defines a parameter to determine the number of layers of subject data. This property must be of a scalar parameter.

② defines the default option, in this case '2'.

③ defines a parameter to determine the labels for each layer. This property must be of string list parameter.

④ defines a parameter to determine the processed labels for each layer. This property must be of string list parameter.

⑤ defines the value from the property 'LAYERLABELS' of SubjectCON\_MP.

⑥ defines the number of layers.

⑥ checks the size of each layer is equal to the number of brain regions.

⑦ defines the height of table.

⑧ defines the option of showing in X-axis slider.

⑨ defines the X-axis sliders' labels.

⑩ defines the option of not showing in Y-axis slider.

**Code 6: SubjectCON\_MP element tests.** The tests section from the element generator \_SubjectCON\_MP.gen.m. ← [Code 3](#)

```

1 %% itests!
2
3 %%% itest!
4 %%% iname!
5 GUI

```

```

6 %%% iprobability!
7 .01
8 %%% icode!
9 im_ba = ImporterBrainAtlasXLS('FILE', 'aal90_atlas.xlsx');
10 ba = im_ba.get('BA');
11
12 gr = Group('SUB_CLASS', 'SubjectCON_MP', 'SUB_DICT', IndexedDictionary('
    IT_CLASS', 'SubjectCON_MP'));
13 for i = 1:1:10
14     sub = SubjectCON_MP( ...
15         'ID', ['SUB CON_MP ' int2str(i)], ...
16         'LABEL', ['Subejct CON_MP ' int2str(i)], ...
17         'NOTES', ['Notes on subject CON_MP ' int2str(i)], ...
18         'BA', ba, ...
19         'L', 3, ... ①
20         'LAYERLABELS', {'L1' 'L2' 'L3'}, ... ②
21         'CON_MP', {rand(ba.get('BR_DICT').get('LENGTH')), rand(ba.get('
    BR_DICT').get('LENGTH')), rand(ba.get('BR_DICT').get('LENGTH'))} ...
22     ); ③
23     sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
        rand()))
24     sub.memorize('VOI_DICT').get('ADD', VOICategorical('ID', 'Sex', '
        CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
25     gr.get('SUB_DICT').get('ADD', sub)
26 end
27
28 gui = GUIElement('PE', gr, 'CLOSEREQ', false);
29 gui.get('DRAW')
30 gui.get('SHOW')
31
32 gui.get('CLOSE')

```

① defines the number of layers.

② defines the label of each layer.

③ constructs 3 layers randomly with size of brain regions by brain regions.

## *Implementation of a subject with functional data*

### *subject with functional data (SubjectFUN)*

We will start by implementing in detail SubjectFUN. The connectivity matrix can be obtained from fMRI data.

**Code 7: SubjectFUN element header.** The header section of the generator code for `_SubjectFUN.gen.m` provides the general information about the SubjectFUN element. [← Code 1](#)

---

```

1
2 %% iheader!
3 SubjectFUN < Subject (sub, subject with functional matrix) is a subject with
   functional matrix (e.g. fMRI).
4
5 %%% idescription!
6 Subject with a functional matrix (e.g. obtained from fMRI).
7
8 %%% iseealso!
9 ImporterGroupSubjectFUN.TXT, ExporterGroupSubjectFUN.TXT,
   ImporterGroupSubjectFUN.XLS, ExporterGroupSubjectFUN.XLS

```

---

**Code 8: SubjectFUN element prop update.** The `props_update` section of the generator code for `_SubjectFUN.gen.m` updates the properties of the Subject element. [← Code 2](#)

---

```

1 %% iprops_update!
2
3 %%% iprop!
4 NAME (constant, string) is the name of the subject.
5 %%% idefault!
6 'SubjectFUN'
7
8 %%% iprop!
9 DESCRIPTION (constant, string) is the description of the subject.
10 %%% idefault!
11 'Subject with a functional matrix (e.g. obtained from fMRI).'
12
13 %%% iprop!
14 TEMPLATE (parameter, item) is the template of the subject.
15 %%% isettings!
16 'SubjectFUN'
17
18 %%% iprop!
19 ID (data, string) is a few-letter code for the subject.
20 %%% idefault!
21 'SubjectFUN ID'
22
23 %%% iprop!
24 LABEL (metadata, string) is an extended label of the subject.
25 %%% idefault!
26 'SubjectFUN label'
27
28 %%% iprop!
29 NOTES (metadata, string) are some specific notes about the subject.
30 %%% idefault!
31 'SubjectFUN notes'

```



```

32
33 %% iprops!
34
35 %%% iprop!
36 BA (data, item) is a brain atlas.
37 %%% isettings!
38 'BrainAtlas'
39
40 %%% iprop!
41 FUN (data, matrix) is an adjacency matrix.
42 %%% icheck_value!
43 br_number = sub.get('BA').get('BR_DICT').get('LENGTH');
44 check = size(value, 2) == br_number; ①
45 if check
46     msg = 'All ok!';
47 else
48     msg = ['FUN must be a matrix with the same number of columns as the
49           brain regions (' int2str(br_number) ').'];
50 end
51 %%% igui! ②
52 pr = PanelPropMatrix('EL', sub, 'PROP', SubjectFUN.FUN, ...
53     'ROWNAME', {'numbered'}, ...
54     'COLUMNNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
55     varargin{:});

```

---

① checks the size of the column of value is equal to the number of brain regions. The rows of value represent the time series.

② Same as in note ④ of Code 2.

Code 9: **SubjectFUN element tests.** The tests section from the element generator `_SubjectFUN.gen.m`. ← [Code 3](#)

---

```

1 %% itests!
2
3 %% itest!
4 %%% iname!
5 GUI
6 %%% iprobability!
7 .01
8 %%% icode!
9 im_ba = ImporterBrainAtlasXLS('FILE', 'aal90_atlas.xlsx');
10 ba = im_ba.get('BA');
11
12 gr = Group('SUB_CLASS', 'SubjectFUN', 'SUB_DICT', IndexedDictionary('
    IT_CLASS', 'SubjectFUN'));
13 for i = 1:1:50
14     sub = SubjectFUN( ...
15         'ID', ['SUB FUN ' int2str(i)], ...
16         'LABEL', ['Subejct FUN ' int2str(i)], ...
17         'NOTES', ['Notes on subject FUN ' int2str(i)], ...
18         'BA', ba, ...
19         'FUN', rand(10, ba.get('BR_DICT').get('LENGTH')) ... ①
20     );
21     sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
        rand()))
22     sub.memorize('VOI_DICT').get('ADD', VOICategorical('ID', 'Sex', '
        CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
23     gr.get('SUB_DICT').get('ADD', sub)
24 end
25
26 gui = GUIElement('PE', gr, 'CLOSEREQ', false);
27 gui.get('DRAW')
28 gui.get('SHOW')
29
30 gui.get('CLOSE')

```

---

① constructs the random adjacency matrix with the size of 10 timepoints by the number of brain regions.

### *subject with functional multiplex data (SubjectFUN\_MP)*

We will start by implementing in detail SubjectFUN\_MP. The functional matrix can be obtained from fMRI data.

**Code 10: SubjectFUN\_MP element header.** The header section of the generator code for `_SubjectFUN_MP.gen.m` provides the general information about the SubjectFUN\_MP element. ← [Code 4](#)

---

```

1
2 %% iheader!
3 SubjectFUN_MP < Subject (sub, subject with functional multiplex data) is a
    subject with functional multiplex data (e.g. multiplex fMRI).
4
5 %%% idescription!
6 Subject with data for each brain region corresponding to L functional layers
    (e.g. activation timeseries obtained from fMRI or EEG).
7
8 %%% iseealso!
9 ImporterGroupSubjectFUN_MP_TXT, ExporterGroupSubjectFUN_MP_TXT,
    ImporterGroupSubjectFUN_MP_XLS, ExporterGroupSubjectFUN_MP_XLS

```

---

**Code 11: SubjectFUN\_MP element prop update.** The `props_update` section of the generator code for `_SubjectFUN_MP.gen.m` updates the properties of the Subject element. This defines the core properties of the Subject. ← [Code 5](#)

---

```

1 %% iprops_update!
2
3 %%% iprop!
4 NAME (constant, string) is the name of the subject.
5 %%% idefault!
6 'SubjectFUN_MP'
7
8 %%% iprop!
9 DESCRIPTION (constant, string) is the description of the subject.
10 %%% idefault!
11 'Subject with data for each brain region corresponding to L functional
    layers (e.g. activation timeseries obtained from fMRI or EEG).'
12
13 %%% iprop!
14 TEMPLATE (parameter, item) is the template of the subject.
15 %%% isettings!
16 'SubjectFUN_MP'
17
18 %%% iprop!
19 ID (data, string) is a few-letter code for the subject.
20 %%% idefault!
21 'SubjectFUN_MP ID'
22
23 %%% iprop!
24 LABEL (metadata, string) is an extended label of the subject.
25 %%% idefault!
26 'SubjectFUN_MP label'
27
28 %%% iprop!
29 NOTES (metadata, string) are some specific notes about the subject.
30 %%% idefault!
31 'SubjectFUN_MP notes'

```

```

32
33 %% iprops!
34
35 %%% iprop!
36 BA (data, item) is a brain atlas.
37 %%%% isettings!
38 'BrainAtlas'
39
40 %%% iprop!
41 L (data, scalar) is the number of layers of subject data. ①
42 %%%% idefault!
43 2
44
45 %%% iprop!
46 LAYERLABELS (metadata, stringlist) are the layer labels provided by the user
   . ②
47
48 %%% iprop!
49 ALAYERLABELS (query, stringlist) returns the processed layer labels. ③
50 %%%% icalculate!
51 value = sub.get('LAYERLABELS');
52
53 %%% iprop!
54 FUN_MP (data, cell) is a cell containing L matrices with each column
   corresponding to the time series of a brain region.
55 %%%% icalculate!
56 br_number = sub.get('BA').get('BR_DICT').get('LENGTH');
57 num_layers = sub.get('L');
58 check = (iscell(value) && isequal(length(value), num_layers) && isequal(
   cellfun(@(v) size(v, 2), value), ones(1, num_layers) * br_number)) || (
   isempty(value) && br_number == 0); ④
59 if check
60     msg = 'All ok!';
61 else
62     msg = ['FUN_MP must be a cell with L matrices with the same number of
   columns as the number of brain regions (' int2str(br_number) ').'];
63 end
64 %%% igui! ⑤
65 pr = PanelPropCell('EL', sub, 'PROP', SubjectFUN_MP.FUN_MP, ...
66     'TABLE_HEIGHT', s(40), ...
67     'XSLIDERSHOW', true, ...
68     'XSLIDERLABELS', sub.getCallback('ALAYERLABELS'), ...
69     'YSLIDERSHOW', false, ...
70     'ROWNAME', {'numbered'}, ...
71     'COLUMNNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
72     varargin{:});

```

① Same as in note ① of Code 5.

② Same as in note ② of Code 5.

③ Same as in note ③ of Code 5.

④ checks the size of each layer are equal to the number of brain regions. The size of each layer is the length of time series by the number of regions.

⑤ Same as in note ⑦ ⑧ ⑨ ⑩ of Code 5.

**Code 12: SubjectFUN\_MP element tests.** The tests section from the element generator `_SubjectFUN_MP.gen.m`. ← Code 6

```

1 %% itests!
2
3 %%% itest!
4 %%%% iname!
5 GUI
6 %%%% iprobability!
7 .01
8 %%%% icode!
9 im_ba = ImporterBrainAtlasXLS('FILE', 'aal90_atlas.xlsx');

```

```

10 ba = im.ba.get('BA');
11
12 gr = Group('SUB_CLASS', 'SubjectFUN_MP', 'SUB_DICT', IndexedDictionary('
    IT_CLASS', 'SubjectFUN_MP'));
13 for i = 1:1:10 ①
14     sub = SubjectFUN_MP( ...
15         'ID', ['SUB FUN_MP ' int2str(i)], ...
16         'LABEL', ['Subejct FUN_MP ' int2str(i)], ...
17         'NOTES', ['Notes on subject FUN_MP ' int2str(i)], ...
18         'BA', ba, ...
19         'L', 3, ...
20         'LAYERLABELS', {'L1' 'L2' 'L3'}, ...
21         'FUN_MP', {rand(10, ba.get('BR_DICT').get('LENGTH')), rand(10, ba.
get('BR_DICT').get('LENGTH')), rand(10, ba.get('BR_DICT').get('LENGTH')
    )} ...
22     );
23 sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
    rand()));
24 sub.memorize('VOI_DICT').get('ADD', VOICategorical('ID', 'Sex', '
    CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
25 gr.get('SUB_DICT').get('ADD', sub)
26 end
27
28 gui = GUIElement('PE', gr, 'CLOSEREQ', false);
29 gui.get('DRAW')
30 gui.get('SHOW')
31
32 gui.get('CLOSE')

```

---

① Same as in note ① ② ③ of Code 6.

## Implementation of a subject with structural data

### subject with structural data (SubjectST)

We will start by implementing in detail SubjectST. The structural matrix can be obtained from sMRI data.

**Code 13: SubjectST element header.** The header section of the generator code for `_SubjectST.gen.m` provides the general information about the SubjectST element. ← [Code 1](#)

---

```

1
2 %% iheader!
3 SubjectST < Subject (sub, subject with structural data) is a subject with
   structural data (e.g. sMRI).
4
5 %%% idescription!
6 Subject with structural data (e.g. cortical thickness obtained from
   strcutural MRI) for each brain region.
7
8 %%% iseealso!
9 ImporterGroupSubjectST_TXT, ExporterGroupSubjectST_TXT,
   ImporterGroupSubjectST_XLS, ExporterGroupSubjectST_XLS

```

---

**Code 14: SubjectST element prop update.** The `props_update` section of the generator code for `_SubjectST.gen.m` updates the properties of the Subject element. ← [Code 2](#)

---

```

1 %% iprops_update!
2
3 %%% iprop!
4 NAME (constant, string) is the name of the subject.
5 %%% idefault!
6 'SubjectST'
7
8 %%% iprop!
9 DESCRIPTION (constant, string) is the description of the subject.
10 %%% idefault!
11 'SubjectST with structural data (e.g. cortical thickness obtained from
   strcutural MRI) for each brain region.'
12
13 %%% iprop!
14 TEMPLATE (parameter, item) is the template of the subject.
15 %%% isettings!
16 'SubjectST'
17
18 %%% iprop!
19 ID (data, string) is a few-letter code for the subject.
20 %%% idefault!
21 'SubjectST ID'
22
23 %%% iprop!
24 LABEL (metadata, string) is an extended label of the subject.
25 %%% idefault!
26 'SubjectST label'
27
28 %%% iprop!
29 NOTES (metadata, string) are some specific notes about the subject.

```

---

```

30 %%%% idefault!
31 'SubjectST notes'
32
33 %% iprops!
34
35 %%% iprop!
36 BA (data, item) is a brain atlas.
37 %%%% isettings!
38 'BrainAtlas'
39
40 %%% iprop!
41 ST (data, cvector) is a column vector with data for each brain region.
42 %%%% icheck_value!
43 br_number = sub.get('BA').get('BR_DICT').get('LENGTH');
44 check = (iscolumn(value) && isequal(size(value), [br_number, 1])) || (
45     isempty(value) && br_number == 0); ①
46 if check
47     msg = 'All ok!';
48 else
49     msg = ['ST must be a column vector with the same number of element as
50         the brain regions (' int2str(br_number) ').'];
51 end
52 %%%% igui! ②
53 pr = PanelPropMatrix('EL', sub, 'PROP', SubjectST.ST, ...
54     'ROWNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
55     'COLUMNNAME', {}, ...
56     varargin{:});

```

---

① checks the size of the row of value is equal to the number of brain regions. The number of column is 1.

② Same as in note ④ of Code 2.

Code 15: **SubjectST element tests.** The tests section from the element generator `_SubjectST.gen.m`. ← [Code 3](#)

---

```

1 %% itests!
2
3 %% itest!
4 %%% iname!
5 GUI
6 %%% iprobability!
7 .01
8 %%% icode!
9 im_ba = ImporterBrainAtlasXLS('FILE', 'destrieux_atlas.xlsx');
10 ba = im_ba.get('BA');
11
12 gr = Group('SUB_CLASS', 'SubjectST', 'SUB_DICT', IndexedDictionary('IT_CLASS
    ', 'SubjectST'));
13 for i = 1:1:50
14     sub = SubjectST( ...
15         'ID', ['SUB ST ' int2str(i)], ...
16         'LABEL', ['Subejct ST ' int2str(i)], ...
17         'NOTES', ['Notes on subject ST ' int2str(i)], ...
18         'BA', ba, ...
19         'ST', rand(ba.get('BR_DICT').get('LENGTH'), 1) ... ①
20     );
21     sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
        rand()))
22     sub.memorize('VOI_DICT').get('ADD', VOICategorical('ID', 'Sex', '
        CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
23     gr.get('SUB_DICT').get('ADD', sub)
24 end
25
26 gui = GUIElement('PE', gr, 'CLOSEREQ', false);
27 gui.get('DRAW')
28 gui.get('SHOW')
29
30 gui.get('CLOSE')

```

---

① constructs the random adjacency matrix with size of the number of brain regions by 1.



### *subject with structural multiplex data (SubjectST\_MP)*

We will start by implementing in detail SubjectST\_MP. The structural matrix can be obtained from sMRI data.

**Code 16: SubjectST\_MP element header.** The header section of the generator code for `_SubjectST_MP.gen.m` provides the general information about the SubjectST\_MP element. ← [Code 4](#)

---

```

1
2 %% iheader!
3 SubjectST_MP < Subject (sub, subject with structural multiplex data) is a
  subject with structural multiplex data (e.g. multiplex sMRI).
4
5 %%% idescription!
6 Subject with data for each brain region corresponding to L structural layers
  (e.g. cortical thickness obtained from structural MRI).
7
8 %%% iseealso!
9 ImporterGroupSubjectST_MP_TXT, ExporterGroupSubjectST_MP_TXT,
  ImporterGroupSubjectST_MP_XLS, ExporterGroupSubjectST_MP_XLS

```

---

### **Code 17: SubjectST\_MP element prop update.**

The `props_update` section of the generator code for `_SubjectST_MP.gen.m` updates the properties of the Subject element. ←

**Code 5**

---

```

1 %% iprops_update!
2
3 %%% iprop!
4 NAME (constant, string) is the name of the subject.
5 %%% idefault!
6 'SubjectST_MP'
7
8 %%% iprop!
9 DESCRIPTION (constant, string) is the description of the subject.
10 %%% idefault!
11 'Subject with data for each brain region corresponding to L structural
  layers (e.g. cortical thickness obtained from structural MRI).'
12
13 %%% iprop!
14 TEMPLATE (parameter, item) is the template of the subject.
15 %%% isettings!
16 'SubjectST_MP'
17
18 %%% iprop!
19 ID (data, string) is a few-letter code for the subject.
20 %%% idefault!
21 'SubjectST_MP ID'
22
23 %%% iprop!
24 LABEL (metadata, string) is an extended label of the subject.
25 %%% idefault!
26 'SubjectST_MP label'
27
28 %%% iprop!
29 NOTES (metadata, string) are some specific notes about the subject.
30 %%% idefault!
31 'SubjectST_MP notes'

```

---

```

32
33 %% iprops!
34
35 %%% iprop!
36 BA (data, item) is a brain atlas.
37 %%%% isettings!
38 'BrainAtlas'
39
40 %%% iprop!
41 L (data, scalar) is the number of layers of subject data. ①
42 %%%% idefault!
43 2
44
45 %%% iprop!
46 LAYERLABELS (metadata, stringlist) are the layer labels provided by the user
   . ②
47
48 %%% iprop!
49 ALAYERLABELS (query, stringlist) returns the processed layer labels. ③
50 %%%% icalculate!
51 value = sub.get('LAYERLABELS');
52
53 %%% iprop!
54 ST_MP (data, cell) is a cell containing L vectors, each with data for each
   brain region.
55 %%%% ickvalue!
56 br_number = sub.get('BA').get('BR_DICT').get('LENGTH');
57 num_layers = sub.get('L');
58 check = (iscell(value) && isequal(length(value), num_layers) && isequal(
   cellfun(@(v) size(v, 1), value), ones(1, num_layers) * br_number)) || (
   isempty(value) && br_number == 0); ④
59 if check
60     msg = 'All ok!';
61 else
62     msg = ['ST_MP must be a column vector with the same number of element as
   the brain regions (' int2str(br_number) ').'];
63 end
64 %%% igui! ⑤
65 pr = PanelPropCell('EL', sub, 'PROP', SubjectST_MP.ST_MP, ...
66     'TABLE_HEIGHT', s(40), ...
67     'XSLIDERSHOW', true, ...
68     'XSLIDERLABELS', sub.getCallback('ALAYERLABELS'), ...
69     'YSLIDERSHOW', false, ...
70     'ROWNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
71     'COLUMNNAME', {}, ...
72     varargin{:});

```

① Same as in note ① of Code 5.

② Same as in note ② of Code 5.

③ Same as in note ③ of Code 5.

④ checks the size of each layer are equal to the number of brain regions. The size of each layer is the number of regions by 1.

⑤ Same as in note ⑦ ⑧ ⑨ ⑩ of Code 5.

**Code 18: SubjectST\_MP element tests.** The tests section from the element generator \_SubjectST\_MP.gen.m. ← Code 6

```

1
2 %% itests!
3
4 %%% itest!
5 %%%% iname!
6 GUI
7 %%%% iprobability!
8 .01
9 %%%% icode!

```

```

10 im_ba = ImporterBrainAtlasXLS('FILE', 'destrieux_atlas.xlsx');
11 ba = im_ba.get('BA');
12
13 gr = Group('SUB_CLASS', 'SubjectST_MP', 'SUB_DICT', IndexedDictionary('
    IT_CLASS', 'SubjectST_MP'));
14 for i = 1:1:10 ①
15     sub = SubjectST_MP( ...
16         'ID', ['SUB ST_MP ' int2str(i)], ...
17         'LABEL', ['Subejct ST_MP ' int2str(i)], ...
18         'NOTES', ['Notes on subject ST_MP ' int2str(i)], ...
19         'BA', ba, ...
20         'L', 3, ...
21         'LAYERLABELS', {'L1' 'L2' 'L3'}, ...
22         'ST_MP', {rand(ba.get('BR_DICT').get('LENGTH'), 1), rand(ba.get('
    BR_DICT').get('LENGTH'), 1), rand(ba.get('BR_DICT').get('LENGTH'), 1)}
    ...
23     );
24     sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
        rand()))
25     sub.memorize('VOI_DICT').get('ADD', VOICategorical('ID', 'Sex', '
        CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
26     gr.get('SUB_DICT').get('ADD', sub)
27 end
28
29 gui = GUIElement('PE', gr, 'CLOSEREQ', false);
30 gui.get('DRAW')
31 gui.get('SHOW')
32
33 gui.get('CLOSE')

```

---

① Same as in note ① ② ③ of Code 6.