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

Implementation of a subject with connectivity matrix

Subject with connectivity data (SubjectCON)

We will start by implementing in detail `SubjectCON`. This class can be used for subjects with connectivity matrix data usually obtained from DTI.

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

```
%% ¡header!
SubjectCON < Subject (sub, subject with connectivity matrix) is a subject with connectivity matrix (e.g. DTI). ①

%%% ¡description!
Subject with a connectivity matrix (e.g. obtained from DTI).

%%% ¡seealso! ②
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.

```
%% ¡props_update!

%%% ¡prop!
NAME (constant, string) is the name of the subject.
%%%% ¡default!
'SubjectCON'

%%% ¡prop!
DESCRIPTION (constant, string) is the description of the subject.
%%%% ¡default!
'SubjectCON with a connectivity matrix (e.g. obtained from DTI).'

%%% ¡prop!
TEMPLATE (parameter, item) is the template of the subject.
%%% ¡settings!
'SubjectCON'

%%% ¡prop!
ID (data, string) is a few-letter code for the subject.
%%%% ¡default!
'SubjectCON ID'

%%% ¡prop!
LABEL (metadata, string) is an extended label of the subject.
%%%% ¡default!
'SubjectCON label'

%%% ¡prop!
NOTES (metadata, string) are some specific notes about the subject.
```

```
%%%% ¡default!
'SubjectCON notes'

%% ¡props!

%%% ¡prop!
BA (data, item) is a brain atlas.
%%%% ¡settings!
'BrainAtlas'


%%% ¡prop!
CON (data, smatrix) is an adjacency matrix.
%%%% ¡check_value!
br_number = sub.get('BA').get('BR_DICT').get('LENGTH'); ①
check = isequal(size(value), [br_number, br_number]);  ②
if check  ③
    msg = 'All ok!';
else
    msg = ['CON must be a square matrix with the dimension equal to the
     number of brain regions (' int2str(br_number) ').'];
end

%%%% ¡gui! ④
pr = PanelPropMatrix('EL', sub, 'PROP', SubjectCON.CON, ...
    'ROWNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
    'COLUMNNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
    varargin{:});
```

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

② checks that 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.

```
%% ¡tests!

%%% ¡test!
%%%% ¡name!
GUI ①
%%%% ¡probability! ②
.01
%%%% ¡code!
im_ba = ImporterBrainAtlasXLS('FILE', 'desikan_atlas.xlsx'); ③
ba = im_ba.get('BA'); ④

gr = Group('SUB_CLASS', 'SubjectCON', 'SUB_DICT', IndexedDictionary('IT_CLASS', 'SubjectCON')); ⑤
for i = 1:1:50 ⑥
    sub = SubjectCON( ... ⑦
        'ID', ['SUB CON ' int2str(i)], ...
        'LABEL', ['Subejct CON ' int2str(i)], ...
        'NOTES', ['Notes on subject CON ' int2str(i)], ...
        'BA', ba, ...
        'CON', rand(ba.get('BR_DICT').get('LENGTH')) ...
        );
    sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 * rand())) ⑧
    sub.memorize('VOI_DICT').get('ADD', VOICategoric('ID', 'Sex', 'CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1))) ⑨
    gr.get('SUB_DICT').get('ADD', sub) ⑩
end

gui = GUIElement('PE', gr, 'CLOSEREQ', false); ⑪
gui.get('DRAW') ⑫
gui.get('SHOW') ⑬

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 Braph2 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 iterates to create 50 subjects.

⑦ creates a new subject of the class `SubjectCON`. For that, it defines the `'ID'`, `'LABEL'`, `'NOTES'`, `'BA'` (Brain Atlas) and `'CON'` (a random adjacency matrix) for the new subject.

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

⑧ adds a random `Categoric` `'Sex'` as the variable of interest of the subject.

⑩ adds the created subject `'sub'` into group of subjects.

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 multilayer 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

```

%% ¡header!
SubjectCON_MP < Subject (sub, subject with connectivity multiplex data) is a
      subject with connectivity multiplex data.

%%% ¡description!
Subject with L connectivity matrices (e.g. obtained from DTI).

%%% ¡seealso!
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

```
%% ¡props_update!

%%% ¡prop!
NAME (constant, string) is the name of the subject.
%%%% ¡default!
'SubjectCON_MP'

%%% ¡prop!
DESCRIPTION (constant, string) is the description of the subject.
%%%% ¡default!
'Subject with L connectivity matrices (e.g. obtained from DTI).'

%%% ¡prop!
TEMPLATE (parameter, item) is the template of the subject.
%%% ¡settings!
'SubjectCON_MP'

%%% ¡prop!
ID (data, string) is a few-letter code for the subject.
%%%% ¡default!
'SubjectCON_MP ID'

%%% ¡prop!
LABEL (metadata, string) is an extended label of the subject.
%%%% ¡default!
'SubjectCON_MP label'

%%% ¡prop!
NOTES (metadata, string) are some specific notes about the subject.
%%%% ¡default!
```

```
'SubjectCON_MP notes'

%% ¡props!

%%% ¡prop!
BA (data, item) is a brain atlas.
%%%% ¡settings!
'BrainAtlas'

%%% ¡prop!
L (data, scalar) is the number of layers of subject data. ①
%%%% ¡default!
2 ②

%%% ¡prop!
LAYERLABELS (metadata, stringlist) are the layer labels provided by the user
     . ③

%%% ¡prop!
ALAYERLABELS (query, stringlist) returns the processed layer labels. ④
%%%% ¡calculate!
value = sub.get('LAYERLABELS'); ⑤

%%% ¡prop!
CON_MP (data, cell) is a cell containing L matrices corresponding
     connectivity matrices of each layer.
%%%% ¡check_value!
br_number = sub.get('BA').get('BR_DICT').get('LENGTH');
num_layers = sub.get('L'); ⑥
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
if check
    msg = 'All ok!';
else
    msg = ['CON_MP must be a cell with L square matrices with the dimension
     equal to the number of brain regions (' int2str(br_number) ').'];
end
%%%% ¡gui!
pr = PanelPropCell('EL', sub, 'PROP', SubjectCON_MP.CON_MP, ...
    'TABLE_HEIGHT', s(40), ... ⑦
    'XSLIDERSHOW', true, ... ⑧
    'XSLIDERLABELS', sub.getCallback('ALAYERLABELS'), ... ⑨
    'YSLIDERSHOW', false, ... ⑩
    'ROWNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
    'COLUMNNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
    varargin{:});
```

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

```
%% ¡tests!

%%% ¡test!
%%%% ¡name!
GUI
```

```
%%%% ¡probability!
.01
%%%% ¡code!
im_ba = ImporterBrainAtlasXLS('FILE', 'aal90_atlas.xlsx');
ba = im_ba.get('BA');

gr = Group('SUB_CLASS', 'SubjectCON_MP', 'SUB_DICT', IndexedDictionary('
     IT_CLASS', 'SubjectCON_MP'));
for i = 1:1:10
    sub = SubjectCON_MP( ...
        'ID', ['SUB CON_MP ' int2str(i)], ...
        'LABEL', ['Subejct CON_MP ' int2str(i)], ...
        'NOTES', ['Notes on subject CON_MP ' int2str(i)], ...
        'BA', ba, ...
        'L', 3, ... ①
        'LAYERLABELS', {'L1' 'L2' 'L3'}, ... ②
        'CON_MP', {rand(ba.get('BR_DICT').get('LENGTH')), rand(ba.get('
     BR_DICT').get('LENGTH')), rand(ba.get('BR_DICT').get('LENGTH'))} ...
        ); ③
    sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
     rand()))
    sub.memorize('VOI_DICT').get('ADD', VOICategoric('ID', 'Sex', '
     CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
    gr.get('SUB_DICT').get('ADD', sub)
end

gui = GUIElement('PE', gr, 'CLOSEREQ', false);
gui.get('DRAW')
gui.get('SHOW')

gui.get('CLOSE')
```

① defines the number of layers.

② defines the label of each layer.

③ constructs 3 layers randomly of the same size as the number of 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

```

%% ¡header!
SubjectFUN < Subject (sub, subject with functional matrix) is a subject with functional matrix (e.g. fMRI).

%%% ¡description!
Subject with a functional matrix (e.g. obtained from fMRI).

%%% ¡seealso!
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

```
%% ¡props_update!

%%% ¡prop!
NAME (constant, string) is the name of the subject.
%%%% ¡default!
'SubjectFUN'

%%% ¡prop!
DESCRIPTION (constant, string) is the description of the subject.
%%%% ¡default!
'Subject with a functional matrix (e.g. obtained from fMRI).'

%%% ¡prop!
TEMPLATE (parameter, item) is the template of the subject.
%%% ¡settings!
'SubjectFUN'

%%% ¡prop!
ID (data, string) is a few-letter code for the subject.
%%%% ¡default!
'SubjectFUN ID'

%%% ¡prop!
LABEL (metadata, string) is an extended label of the subject.
%%%% ¡default!
'SubjectFUN label'

%%% ¡prop!
NOTES (metadata, string) are some specific notes about the subject.
%%%% ¡default!
'SubjectFUN notes'
```

```

%% ¡props!

%%% ¡prop!
BA (data, item) is a brain atlas.
%%%% ¡settings!
'BrainAtlas'

%%% ¡prop!
FUN (data, matrix) is an adjacency matrix.
%%%% ¡check_value!
br_number = sub.get('BA').get('BR_DICT').get('LENGTH');
check = size(value, 2) == br_number; ①
if check
    msg = 'All ok!';
else
    msg = ['FUN must be a matrix with the same number of columns as the
     brain regions (' int2str(br_number) ').'];
end
%%%% ¡gui! ②
pr = PanelPropMatrix('EL', sub, 'PROP', SubjectFUN.FUN, ...
    'ROWNAME', {'numbered'}, ...
    'COLUMNNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
    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

```
%% ¡tests!

%%% ¡test!
%%%% ¡name!
GUI
%%%% ¡probability!
.01
%%%% ¡code!
im_ba = ImporterBrainAtlasXLS('FILE', 'aal90_atlas.xlsx');
ba = im_ba.get('BA');

gr = Group('SUB_CLASS', 'SubjectFUN', 'SUB_DICT', IndexedDictionary('
     IT_CLASS', 'SubjectFUN'));
for i = 1:1:50
    sub = SubjectFUN( ...
        'ID', ['SUB FUN ' int2str(i)], ...
        'LABEL', ['Subejct FUN ' int2str(i)], ...
        'NOTES', ['Notes on subject FUN ' int2str(i)], ...
        'BA', ba, ...
        'FUN', rand(10, ba.get('BR_DICT').get('LENGTH')) ...(1)
        );
    sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
     rand()))
    sub.memorize('VOI_DICT').get('ADD', VOICategoric('ID', 'Sex', '
     CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
    gr.get('SUB_DICT').get('ADD', sub)
end

gui = GUIElement('PE', gr, 'CLOSEREQ', false);
gui.get('DRAW')
gui.get('SHOW')

gui.get('CLOSE')
```

(1) constructs the random adjacency matrix with a 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

```

%% ¡header!
SubjectFUN_MP < Subject (sub, subject with functional multiplex data) is a
     subject with functional multiplex data (e.g. multiplex fMRI).

%%% ¡description!
Subject with data for each brain region corresponding to L functional layers
      (e.g. activation timeseries obtaiend from fMRI or EEG).

%%% ¡seealso!
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

```
%% ¡props_update!

%%% ¡prop!
NAME (constant, string) is the name of the subject.
%%%% ¡default!
'SubjectFUN_MP'

%%% ¡prop!
DESCRIPTION (constant, string) is the description of the subject.
%%%% ¡default!
'Subject with data for each brain region corresponding to L functional
     layers (e.g. activation timeseries obtaiend from fMRI or EEG).'

%%% ¡prop!
TEMPLATE (parameter, item) is the template of the subject.
%%% ¡settings!
'SubjectFUN_MP'

%%% ¡prop!
ID (data, string) is a few-letter code for the subject.
%%%% ¡default!
'SubjectFUN_MP ID'

%%% ¡prop!
LABEL (metadata, string) is an extended label of the subject.
%%%% ¡default!
'SubjectFUN_MP label'

%%% ¡prop!
NOTES (metadata, string) are some specific notes about the subject.
%%%% ¡default!
'SubjectFUN_MP notes'
```

```

%% ¡props!

%%% ¡prop!
BA (data, item) is a brain atlas.
%%%% ¡settings!
'BrainAtlas'

%%% ¡prop!
L (data, scalar) is the number of layers of subject data. ①
%%%% ¡default!
2

%%% ¡prop!
LAYERLABELS (metadata, stringlist) are the layer labels provided by the user
    . ②

%%% ¡prop!
ALAYERLABELS (query, stringlist) returns the processed layer labels. ③
%%%% ¡calculate!
value = sub.get('LAYERLABELS');

%%% ¡prop!
FUN_MP (data, cell) is a cell containing L matrices with each column
    corresponding to the time series of a brain region.
%%%% ¡check_value!
br_number = sub.get('BA').get('BR_DICT').get('LENGTH');
num_layers = sub.get('L');
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
if check
    msg = 'All ok!';
else
    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) ').'];
end
%%%% ¡gui! ⑤
pr = PanelPropCell('EL', sub, 'PROP', SubjectFUN_MP.FUN_MP, ...
    'TABLE_HEIGHT', s(40), ...
    'XSLIDERSHOW', true, ...
    'XSLIDERLABELS', sub.getCallback('ALAYERLABELS'), ...
    'YSLIDERSHOW', false, ...
    'ROWNAME', {'numbered'}, ...
    'COLUMNNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
    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 that the size of each layer is 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

```
%% ¡tests!

%%% ¡test!
%%%% ¡name!
GUI
%%%% ¡probability!
.01
%%%% ¡code!
im_ba = ImporterBrainAtlasXLS('FILE', 'aal90_atlas.xlsx');
```

```
ba = im_ba.get('BA');

gr = Group('SUB_CLASS', 'SubjectFUN_MP', 'SUB_DICT', IndexedDictionary('
     IT_CLASS', 'SubjectFUN_MP'));
for i = 1:1:10 ①
    sub = SubjectFUN_MP( ...
        'ID', ['SUB FUN_MP ' int2str(i)], ...
        'LABEL', ['Subejct FUN_MP ' int2str(i)], ...
        'NOTES', ['Notes on subject FUN_MP ' int2str(i)], ...
        'BA', ba, ...
        'L', 3, ...
        'LAYERLABELS', {'L1' 'L2' 'L3'}, ...
        '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')
     )} ...
        );
    sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
     rand()))
    sub.memorize('VOI_DICT').get('ADD', VOICategoric('ID', 'Sex', '
     CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
    gr.get('SUB_DICT').get('ADD', sub)
end

gui = GUIElement('PE', gr, 'CLOSEREQ', false);
gui.get('DRAW')
gui.get('SHOW')

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

```

%% ¡header!
SubjectST < Subject (sub, subject with structural data) is a subject with
     structural data (e.g. sMRI).

%%% ¡description!
Subject with structural data (e.g. cortical thickness obtaibed from
     strcutural MRI) for each brain region.

%%% ¡seealso!
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

```
%% ¡props_update!

%%% ¡prop!
NAME (constant, string) is the name of the subject.
%%%% ¡default!
'SubjectST'

%%% ¡prop!
DESCRIPTION (constant, string) is the description of the subject.
%%%% ¡default!
'SubjectST with structural data (e.g. cortical thickness obtaibed from
     strcutural MRI) for each brain region.'

%%% ¡prop!
TEMPLATE (parameter, item) is the template of the subject.
%%% ¡settings!
'SubjectST'

%%% ¡prop!
ID (data, string) is a few-letter code for the subject.
%%%% ¡default!
'SubjectST ID'

%%% ¡prop!
LABEL (metadata, string) is an extended label of the subject.
%%%% ¡default!
'SubjectST label'

%%% ¡prop!
NOTES (metadata, string) are some specific notes about the subject.
```

```
%%%% ¡default!
'SubjectST notes'

%% ¡props!

%%% ¡prop!
BA (data, item) is a brain atlas.
%%%% ¡settings!
'BrainAtlas'

%%% ¡prop!
ST (data, cvector) is a column vector with data for each brain region.
%%%% ¡check_value!
br_number = sub.get('BA').get('BR_DICT').get('LENGTH');
check = (iscolumn(value) && isequal(size(value), [br_number, 1])) || (
     isempty(value) && br_number == 0); (1)
if check
    msg = 'All ok!';
else
    msg = ['ST must be a column vector with the same number of element as
     the brain regions (' int2str(br_number) ').'];
end
%%%% ¡gui! (2)
pr = PanelPropMatrix('EL', sub, 'PROP', SubjectST.ST, ...
    'ROWNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
    'COLUMNNAME', {}, ...
    varargin{:});
```

(1) checks that the size of the row of `value` is equal to the number of brain regions. The number of columns 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

```
%% ¡tests!

%%% ¡test!
%%%% ¡name!
GUI
%%%% ¡probability!
.01
%%%% ¡code!
im_ba = ImporterBrainAtlasXLS('FILE', 'destrieux_atlas.xlsx');
ba = im_ba.get('BA');

gr = Group('SUB_CLASS', 'SubjectST', 'SUB_DICT', IndexedDictionary('IT_CLASS
     ', 'SubjectST'));
for i = 1:1:50
    sub = SubjectST( ...
        'ID', ['SUB ST ' int2str(i)], ...
        'LABEL', ['Subejct ST ' int2str(i)], ...
        'NOTES', ['Notes on subject ST ' int2str(i)], ...
        'BA', ba, ...
        'ST', rand(ba.get('BR_DICT').get('LENGTH'), 1) ... (1)
        );
    sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
     rand()))
    sub.memorize('VOI_DICT').get('ADD', VOICategoric('ID', 'Sex', '
     CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
    gr.get('SUB_DICT').get('ADD', sub)
end

gui = GUIElement('PE', gr, 'CLOSEREQ', false);
gui.get('DRAW')
gui.get('SHOW')

gui.get('CLOSE')
```

(1) constructs the random adjacency matrix with a size of 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

```

%% ¡header!
SubjectST_MP < Subject (sub, subject with structural multiplex data) is a
     subject with structural multiplex data (e.g. multiplex sMRI).

%%% ¡description!
Subject with data for each brain region corresponding to L structural layers
      (e.g. cortical thickness obtained from structural MRI).

%%% ¡seealso!
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

```
%% ¡props_update!

%%% ¡prop!
NAME (constant, string) is the name of the subject.
%%%% ¡default!
'SubjectST_MP'

%%% ¡prop!
DESCRIPTION (constant, string) is the description of the subject.
%%%% ¡default!
'Subject with data for each brain region correspponding to L structural
     layers (e.g. cortical thickness obtained from structural MRI).'

%%% ¡prop!
TEMPLATE (parameter, item) is the template of the subject.
%%% ¡settings!
'SubjectST_MP'

%%% ¡prop!
ID (data, string) is a few-letter code for the subject.
%%%% ¡default!
'SubjectST_MP ID'

%%% ¡prop!
LABEL (metadata, string) is an extended label of the subject.
%%%% ¡default!
'SubjectST_MP label'

%%% ¡prop!
NOTES (metadata, string) are some specific notes about the subject.
%%%% ¡default!
'SubjectST_MP notes'

```

```
%% ¡props!

%%% ¡prop!
BA (data, item) is a brain atlas.
%%%% ¡settings!
'BrainAtlas'

%%% ¡prop!
L (data, scalar) is the number of layers of subject data. ①
%%%% ¡default!
2

%%% ¡prop!
LAYERLABELS (metadata, stringlist) are the layer labels provided by the user
     . ②

%%% ¡prop!
ALAYERLABELS (query, stringlist) returns the processed layer labels. ③
%%%% ¡calculate!
value = sub.get('LAYERLABELS');

%%% ¡prop!
ST_MP (data, cell) is a cell containing L vectors, each with data for each
     brain region.
%%%% ¡check_value!
br_number = sub.get('BA').get('BR_DICT').get('LENGTH');
num_layers = sub.get('L');
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
if check
    msg = 'All ok!';
else
    msg = ['ST_MP must be a column vector with the same number of element as
      the brain regions (' int2str(br_number) ').'];
end
%%%% ¡gui! ⑤
pr = PanelPropCell('EL', sub, 'PROP', SubjectST_MP.ST_MP, ...
    'TABLE_HEIGHT', s(40), ...
    'XSLIDERSHOW', true, ...
    'XSLIDERLABELS', sub.getCallback('ALAYERLABELS'), ...
    'YSLIDERSHOW', false, ...
    'ROWNAME', sub.get('BA').get('BR_DICT').getCallback('KEYS'), ...
    'COLUMNNAME', {}, ...
    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 is 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

```

%% ¡tests!

%%% ¡test!
%%%% ¡name!
GUI
%%%% ¡probability!
.01
%%%% ¡code!
im_ba = ImporterBrainAtlasXLS('FILE', 'destrieux_atlas.xlsx');
```

```
ba = im_ba.get('BA');

gr = Group('SUB_CLASS', 'SubjectST_MP', 'SUB_DICT', IndexedDictionary('IT_CLASS', 'SubjectST_MP'));
for i = 1:1:10 ①
    sub = SubjectST_MP( ...
        'ID', ['SUB ST_MP ' int2str(i)], ...
        'LABEL', ['Subejct ST_MP ' int2str(i)], ...
        'NOTES', ['Notes on subject ST_MP ' int2str(i)], ...
        'BA', ba, ...
        'L', 3, ...
        'LAYERLABELS', {'L1' 'L2' 'L3'}, ...
        '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)} ...
        );
    sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 * rand()))
    sub.memorize('VOI_DICT').get('ADD', VOICategoric('ID', 'Sex', 'CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
    gr.get('SUB_DICT').get('ADD', sub)
end

gui = GUIElement('PE', gr, 'CLOSEREQ', false);
gui.get('DRAW')
gui.get('SHOW')

gui.get('CLOSE')
```

① Same as in note ① ② ③ of Code 6.