

# *Implement a new Group Analysis*

*The BRAPH 2 Developers*

*December 24, 2024*

This is the developer tutorial for implementing a new group analysis. In this tutorial, you will learn how to create a `*.gen.m` for a new group analysis, which can then be compiled by `braph2genesis`. Here, you will use as examples the group analysis `AnalyzeGroup_ST_BUD`, an group-based graph analysis (`AnalyzeGroup`) analyzing structure data (ST) using binary undirected multigraphs at fixed densities (BUD).

## *Contents*

<i>Implementation of the group analysis</i>	2
<i>Basic properties</i>	2
<i>Functionality-focused properties</i>	3
<i>Verification through testing</i>	5

## Implementation of the group analysis

You will implement in detail `AnalyzeGroup_ST_BUD`, a direct extension of `AnalyzeGroup`. An `AnalyzeGroup_ST_BUD` processes structural data to construct binary undirected graphs at fixed densities.

### Basic properties

This section focuses on implementing the basic properties required to define `AnalyzeGroup_ST_BUD`, including its class, name, and associated metadata.

**Code 1: `AnalyzeGroup_ST_BUD` element header.** The header section of the generator code in `_AnalyzeGroup_ST_BUD.gen.m` provides the general information about the `AnalyzeGroup_ST_BUD` element.

---

```

1 %% iheader! ①
2 AnalyzeGroup_ST_BUD < AnalyzeGroup (a, graph analysis with structural data
   at fixed density) is a graph analysis using structural data at fixed
   density.
3
4 %% idescription! ②
5 AnalyzeGroup_ST_BUD uses structural data at fixed density and analyzes them
   using binary undirected graphs.
6
7 %% iseealso!
8 SubjectST, MultigraphBUD
9
10 %% ibuild! ③
11 1

```

---

① defines `AnalyzeGroup_ST_BUD` as a subclass of `AnalyzeGroup`. The moniker is `a`.

② provides a description of this group analysis.

③ defines the build number of the group analysis element.

**Code 2: Basic properties of `AnalyzeGroup_ST_BUD`.** This section of the generator code in `_AnalyzeGroup_ST_BUD.gen.m` updates the basic properties required to describe the `AnalyzeGroup_ST_BUD` element, including its class, name, description, and other metadata.

---

```

1 %% iprops_update!
2
3 %% iprop!
4 ELCLASS (constant, string) is the class of the group-based graph analysis
   with structural data at fixed density.
5 %%%% idefault!
6 'AnalyzeGroup_ST_BUD'
7
8 %% iprop!
9 NAME (constant, string) is the name of the group-based graph analysis with
   structural data at fixed density.
10 %%%% idefault!
11 'Structural Binary Undirected at fixed Densities Analyze Group'
12
13 %% iprop!
14 DESCRIPTION (constant, string) is the description of the group-based graph
   analysis with structural data at fixed density.
15 %%%% idefault!

```

---

```

16 'AnalyzeGroup_ST_BUD uses structural data at fixed density and analyzes them
    using binary undirected graphs.'

```

---

### *Functionality-focused properties*

This section details the implementation of functionality-focused properties that directly enable `AnalyzeGroup_ST_BUD` to perform graph analysis.

#### **Code 3: Implementation properties of `AnalyzeGroup_ST_BUD`.**

This section of the generator code in `_AnalyzeGroup_ST_BUD.gen.m` updates the properties to be used, including `TEMPLATE` for specifying its `G` with graph type and parameters as a graph template, `GR` for defining subjects' data, and `G` for managing the graph instance obtained using the subjects' data.

```

1
2 %% iprop! ①
3
4 TEMPLATE (parameter, item) is the template of the group-based graph analysis
    with structural data at fixed density.
5 %%%% isettings!
6 'AnalyzeGroup_ST_BUD'
7
8 %% iprop!
9 ID (data, string) is a few-letter code for the group-based graph analysis
    with structural data at fixed density.
10 %%%% idefault!
11 'AnalyzeGroup_ST_BUD ID'
12
13 %% iprop!
14 LABEL (metadata, string) is an extended label of the group-based graph
    analysis with structural data at fixed density.
15 %%%% idefault!
16 'AnalyzeGroup_ST_BUD label'
17
18 %% iprop!
19 NOTES (metadata, string) are some specific notes about the group-based graph
    analysis with structural data at fixed density.
20 %%%% idefault!
21 'AnalyzeGroup_ST_BUD notes'
22
23 %% iprop! ②
24 GR (data, item) is the subject group, which also defines the subject class
    SubjectST.
25 %%%% idefault!
26 Group('SUB_CLASS', 'SubjectST')
27
28 %% iprop! ③
29 G (result, item) is the graph obtained from this analysis.
30 %%%% isettings!
31 'MultigraphBUD'
32 %%%% icalculate!
33 gr = a.get('GR');
34 data_list = cellfun(@(x) x.get('ST'), gr.get('SUB_DICT').get('IT_LIST'), '
    UniformOutput', false); ④

```

① Specifies the `TEMPLATE` property, in which `G`, serving as a graph template, defines parameters such as `DENSITIES`, `SEMIPOSITIVIZE_RULE`, and `STANDARDIZE_RULE`. These settings are applied to the graph in ⑧.

② defines the `GR` property, which stores the subjects using the `SubjectST` element. This property contains the subjects' data to be analyzed.

③ creates the `G` property, a graph that uses `MultigraphBUD`.

④ retrieves the subjects' structural data from ②, which is used to construct the `MultigraphBUD` instance for analysis.

```

35 data = cat(2, data_list{:})'; % correlation is a column based operation
36
37 if any(strcmp(a.get('CORRELATION_RULE'), {Correlation.PEARSON_CV,
38     Correlation.SPEARMAN_CV}))
39     A = Correlation.getAdjacencyMatrix(data, a.get('CORRELATION_RULE'), a.get(
40         'NEGATIVE_WEIGHT_RULE'), gr.get('COVARIATES')); (5)
41 else
42     A = Correlation.getAdjacencyMatrix(data, a.get('CORRELATION_RULE'), a.
43         get('NEGATIVE_WEIGHT_RULE')); (6)
44 end
45
46 densities = a.get('DENSITIES'); % this is a vector (7)
47
48 g = MultigraphBUD( ... (8)
49     'ID', ['Graph ' gr.get('ID')], ...
50     'B', A, ...
51     'DENSITIES', densities, ...
52     'LAYERLABELS', cellfun(@(x) [num2str(x) '%'], num2cell(densities), '
53         UniformOutput', false) ...
54 );
55
56 if ~isa(a.get('TEMPLATE'), 'NoValue') % the analysis has a template
57     g.set('TEMPLATE', a.get('TEMPLATE').memorize('G')) % the template is
58     memorized - overwrite densities (9)
59 end
60
61 if a.get('GR').get('SUB_DICT').get('LENGTH')
62     g.set('NODELABELS', a.get('GR').get('SUB_DICT').get('IT', 1).get('BA').
63         get('BR_DICT').get('KEYS'))
64 end
65
66 value = g;

```

Code 4: **AnalyzeGroup\_ST\_BUD** element props. The props section of the generator code in `_AnalyzeGroup_ST_BUD.gen.m` defines the properties to be used in `AnalyzeGroup_ST_BUD`.

```

1 %% iprops!
2
3 %% iprop! (1)
4 CORRELATION_RULE (parameter, option) is the correlation type.
5 %%% isettings!
6 Correlation.CORRELATION_RULE_LIST
7 %%% idefault!
8 Correlation.PEARSON
9
10 %% iprop! (2)
11 NEGATIVE_WEIGHT_RULE (parameter, option) determines how to deal with
12     negative weights.
13 %%% isettings!
14 Correlation.NEGATIVE_WEIGHT_RULE_LIST
15 %%% idefault!
16 Correlation.ZERO
17
18 %% iprop! (3)
19 DENSITIES (parameter, rvector) is the vector of densities.
20 %%% idefault!
21 [1:1:10]

```

(5) or (6) calculates the adjacency matrix based on whether group covariates (e.g., age and sex) are considered. Covariates are included for partial correlation if the `CORRELATION_RULE` property, introduced in (1) of Code 4, is set to `PEARSON_CV` (Pearson with covariates) or `SPEARMAN_CV` (Spearman with covariates).

(7) retrieves the densities defined in the new property in (2) of Code 4. These densities configure the `MultigraphBUD` instance for analysis.

(8) defines the graph by constructing an instance of `MultigraphBUD` for the calculated adjacency matrix and applying the specified `DENSITIES` parameter.

(9) ensures the `MultigraphBUD` instance is updated with pre-defined parameters (e.g., `DENSITIES`, `SEMIPOSITIVIZE_RULE`, and `STANDARDIZE_RULE`) from the graph template specified in (1). If explicitly set by the user during initialization of `AnalyzeGroup_ST_BUD`, the densities will be overwritten with those in the graph template.

(1) defines the `CORRELATION_RULE` as one of the options in the list (`PEARSON`, `SPEARMAN`, `KENDALL`, `PEARSON_CV`, `SPEARMAN_CV`), used in (5) of Code 3.

(2) defines the `NEGATIVE_WEIGHT_RULE` as one of the options in the list (`ZERO`, `ABS`, `NONE`), used in (5) of Code 3.

(3) defines the densities for binarizing the connectivity matrix in a binary undirected multigraph, used in (7) of Code 3.

```

21 %%% igui! ④
22 pr = PanelPropRVectorSmart('EL', a, 'PROP', AnalyzeGroup_ST_BUD.DENSITIES,
    ...
23 'MIN', 0, 'MAX', 100, ...
24 'DEFAULT', AnalyzeGroup_ST_BUD.getPropDefault('DENSITIES'), ...
25 varargin{:});

```

④ PanelPropRVectorSmart renders a GUI row vector panel for defining densities, supporting MATLAB expressions and limiting values between MIN and MAX.

### Verification through testing

This section tests `AnalyzeGroup_ST_BUD` to confirm its functionality via example scripts and ensure GUI integration.

**Code 5: `AnalyzeGroup_ST_BUD` element tests.** The tests section in the element generator `_AnalyzeGroup_ST_BUD.gen.m` includes logic test, which verifies correct functionality using example scripts and simulated datasets, and integration tests, which ensure the instance operation of the direct GUI and associated GUIs.

```

1
2 %% itests!
3
4 %%% itest! ①
5 %%% iname!
6 Example
7 %%% iprobability! ②
8 .01
9 %%% icode!
10 create_data_ST_XLS() % only creates files if the example folder doesn't
    already exist
11
12 example_ST_BUD
13
14 %%% itest! ③
15 %%% iname!
16 GUI - Analysis
17 %%% iprobability!
18 .01
19 %%% icode!
20 im_ba = ImporterBrainAtlasXLS('FILE', 'destrieux_atlas.xlsx');
21 ba = im_ba.get('BA'); ④
22
23 gr = Group('SUB_CLASS', 'SubjectST', 'SUB_DICT', IndexedDictionary('IT_CLASS',
    'SubjectST')); ⑤
24 for i = 1:1:50
25     sub = SubjectST( ...
26         'ID', ['SUB ST ' int2str(i)], ...
27         'LABEL', ['Subject ST ' int2str(i)], ...
28         'NOTES', ['Notes on subject ST ' int2str(i)], ...
29         'BA', ba, ...
30         'ST', rand(ba.get('BR_DICT').get('LENGTH'), 1) ...
31     );
32     sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
        rand()))
33     sub.memorize('VOI_DICT').get('ADD', VOICategorical('ID', 'Sex', '
        CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
34     gr.get('SUB_DICT').get('ADD', sub)

```

① Tests the functionality of `AnalyzeGroup_ST_BUD` using an example script.

② assigns a low test execution probability.

③ Tests the direct GUI functionality of `AnalyzeGroup_ST_BUD`.

④ and ⑤ define the necessary objects required to initialize an instance of `AnalyzeGroup_ST_BUD`.

```

35 end
36
37 a = AnalyzeGroup_ST_BUD('GR', gr, 'DENSITIES', 5:10:35); (6)
38
39 gui = GUIElement('PE', a, 'CLOSEREQ', false); (7)
40 gui.get('DRAW') (8)
41 gui.get('SHOW') (9)
42
43 gui.get('CLOSE') (10)
44
45 %% itest! (11)
46 %%% iname!
47 GUI - Comparison
48 %%% iprobability!
49 .01
50 %%% icode!
51 im_ba = ImporterBrainAtlasXLS('FILE', 'destrieux_atlas.xlsx');
52 ba = im_ba.get('BA');
53
54 gr1 = Group('SUB_CLASS', 'SubjectST', 'SUB_DICT', IndexedDictionary('
    IT_CLASS', 'SubjectST'));
55 for i = 1:1:50
56     sub = SubjectST( ...
57         'ID', ['SUB ST ' int2str(i)], ...
58         'LABEL', ['Subject ST ' int2str(i)], ...
59         'NOTES', ['Notes on subject ST ' int2str(i)], ...
60         'BA', ba, ...
61         'ST', rand(ba.get('BR_DICT').get('LENGTH'), 1) ...
62     );
63     sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
        rand()))
64     sub.memorize('VOI_DICT').get('ADD', VOICategorical('ID', 'Sex', '
        CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
65     gr1.get('SUB_DICT').get('ADD', sub)
66 end
67
68 gr2 = Group('SUB_CLASS', 'SubjectST', 'SUB_DICT', IndexedDictionary('
    IT_CLASS', 'SubjectST'));
69 for i = 1:1:50
70     sub = SubjectST( ...
71         'ID', ['SUB ST ' int2str(i)], ...
72         'LABEL', ['Subject ST ' int2str(i)], ...
73         'NOTES', ['Notes on subject ST ' int2str(i)], ...
74         'BA', ba, ...
75         'ST', rand(ba.get('BR_DICT').get('LENGTH'), 1) ...
76     );
77     sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
        rand()))
78     sub.memorize('VOI_DICT').get('ADD', VOICategorical('ID', 'Sex', '
        CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
79     gr2.get('SUB_DICT').get('ADD', sub)
80 end
81
82 a1 = AnalyzeGroup_ST_BUD('GR', gr1, 'DENSITIES', 5:10:35); (12)
83 a2 = AnalyzeGroup_ST_BUD('GR', gr2, 'TEMPLATE', a1); (13)
84
85 c = CompareGroup( ... (14)

```

(6) initializes an AnalyzeGroup\_ST\_BUD instance using the specified gr (group) and densities.

(7), (8), and (9) test the process of creating a GUI for AnalyzeGroup\_ST\_BUD, drawing it, and showing it on the screen.

(10) tests the process of closing the shown GUI.

(11) tests the associated GUI functionality of AnalyzeGroup\_ST\_BUD.

(12) Similar to the previous test, this initializes the first AnalyzeGroup\_ST\_BUD with the specified gr and densities.

(13) initializes the second AnalyzeGroup\_ST\_BUD using the first AnalyzeGroup\_ST\_BUD instance as a template. This setup allows the second instance to have its own gr data while applying the same parameters, specifically the densities.

(14) creates a CompareGroup instance with the defined AnalyzeGroup\_ST\_BUD instances.

```
86     'P', 10, ...
87     'A1', a1, ...
88     'A2', a2, ...
89     'WAITBAR', true, ...
90     'VERBOSE', false, ...
91     'MEMORIZE', true ...
92 );
93
94 gui = GUIElement('PE', c, 'CLOSEREQ', false); (15)
95 gui.get('DRAW') (16)
96 gui.get('SHOW') (17)
97
98 gui.get('CLOSE') (18)
```

---

(15), (16), (17), and (18) test creating, drawing, showing, and closing the GUI of the CompareGroup, which is an associated GUI of AnalyzeGroup\_ST\_BUD