Implement a New Group Analysis

The BRAPH 2 Developers

January 4, 2025

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`, a group-based graph analysis (AnalyzeGroup) analyzing structural data (ST) using binary undirected multigraphs at fixed densities (BUD).

Contents

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.

```
%% iheader! (1)
AnalyzeGroup_ST_BUD < AnalyzeGroup (a, graph analysis with structural data
     at fixed density) is a graph analysis using structural data at fixed
     density.

%%% idescription! (2)
AnalyzeGroup_ST_BUD uses structural data at fixed density and analyzes them
     using binary undirected graphs.

%%% iseealso!
SubjectST, MultigraphBUD

%%% ibuild! (3)
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.

```
%% iprops_update!

%%% iprop!
ELCLASS (constant, string) is the class of the group-based graph analysis
     with structural data at fixed density.
%%%% idefault!
'AnalyzeGroup_ST_BUD'

%%% iprop!
NAME (constant, string) is the name of the group-based graph analysis with
     structural data at fixed density.
%%%% idefault!
'Structural Binary Undirected at fixed Densities Analyze Group'

%%% iprop!
DESCRIPTION (constant, string) is the description of the group-based graph
     analysis with structural data at fixed density.
%%%% idefault!
```

```
'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.

```

%%% iprop! (1)

TEMPLATE (parameter, item) is the template of the group-based graph analysis
    with structural data at fixed density.
%%%% isettings!
'AnalyzeGroup_ST_BUD'

%%% iprop!
ID (data, string) is a few-letter code for the group-based graph analysis
    with structural data at fixed density.
%%%% idefault!
'AnalyzeGroup_ST_BUD ID'

%%% iprop!
LABEL (metadata, string) is an extended label of the group-based graph
    analysis with structural data at fixed density.
%%%% idefault!
'AnalyzeGroup_ST_BUD label'

%%% iprop!
NOTES (metadata, string) are some specific notes about the group-based graph
    analysis with structural data at fixed density.
%%%% idefault!
'AnalyzeGroup_ST_BUD notes'

%%% iprop! (2)
GR (data, item) is the subject group, which also defines the subject class
    SubjectST.
%%%% idefault!
Group('SUB_CLASS', 'SubjectST')

%%% iprop! (3)
G (result, item) is the graph obtained from this analysis.
%%%% isettings!
'MultigraphBUD'
%%%% icalculate!
gr = a.get('GR');
data_list = cellfun(@(x) x.get('ST'), gr.get('SUB_DICT').get('IT_LIST'), '
    UniformOutput', false); (4)
```

(1) Sspecifies 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.

```
data = cat(2, data_list{:})'; % correlation is a column based operation

if any(strcmp(a.get('CORRELATION_RULE'), {Correlation.PEARSON_CV,
    Correlation.SPEARMAN_CV}))
  A = Correlation.getAdjacencyMatrix(data, a.get('CORRELATION_RULE'), a.get(
    'NEGATIVE_WEIGHT_RULE'), gr.get('COVARIATES'));(5)
else
    A = Correlation.getAdjacencyMatrix(data, a.get('CORRELATION_RULE'), a.
    get('NEGATIVE_WEIGHT_RULE'));(6)
end

densities = a.get('DENSITIES'); % this is a vector (7)

g = MultigraphBUD( ...(8)
    'ID', ['Graph ' gr.get('ID')], ...
    'B', A, ...
    'DENSITIES', densities, ...
    'LAYERLABELS', cellfun(@(x) [num2str(x) '%'], num2cell(densities), '
    UniformOutput', false) ...
    );

if ~isa(a.getr('TEMPLATE'), 'NoValue') % the analysis has a template
    g.set('TEMPLATE', a.get('TEMPLATE').memorize('G')) % the template is
    memorized - overwrite densities (9)
end

if a.get('GR').get('SUB_DICT').get('LENGTH')
    g.set('NODELABELS', a.get('GR').get('SUB_DICT').get('IT', 1).get('BA').
    get('BR_DICT').get('KEYS'))
end

value = g;
```

(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.

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`.

```
%% iprops!

%%% iprop! (1)
CORRELATION_RULE (parameter, option) is the correlation type.
%%%% isettings!
Correlation.CORRELATION_RULE_LIST
%%%% idefault!
Correlation.PEARSON

%%% iprop! (2)
NEGATIVE_WEIGHT_RULE (parameter, option) determines how to deal with
    negative weights.
%%%% isettings!
Correlation.NEGATIVE_WEIGHT_RULE_LIST
%%%% idefault!
Correlation.ZERO

%%% iprop! (3)
DENSITIES (parameter, rvector) is the vector of densities.
%%%% idefault!
[1:1:10]
```

(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.

```
%%%% igui! (4)
pr = PanelPropRVectorSmart('EL', a, 'PROP', AnalyzeGroup_ST_BUD.DENSITIES,
     ...
    'MIN', 0, 'MAX', 100, ...
    'DEFAULT', AnalyzeGroup_ST_BUD.getPropDefault('DENSITIES'), ...
    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.

```

%% itests!

%%% itest!(1)
%%%% iname!
Example
%%%% iprobability!(2)
.01
%%%% icode!
create_data_ST_XLS() % only creates files if the example folder doesn't
     already exist

example_ST_BUD

%%% itest! (3)
%%%% iname!
GUI - Analysis
%%%% iprobability!
.01
%%%% icode!
im_ba = ImporterBrainAtlasXLS('FILE', 'destrieux_atlas.xlsx');
ba = im_ba.get('BA');(4)

gr = Group('SUB_CLASS', 'SubjectST', 'SUB_DICT', IndexedDictionary('IT_CLASS
     ', 'SubjectST')); (5)
for i = 1:1:50
    sub = SubjectST( ...
        'ID', ['SUB ST ' int2str(i)], ...
        'LABEL', ['Subject ST ' int2str(i)], ...
        'NOTES', ['Notes on subject ST ' int2str(i)], ...
        'BA', ba, ...
        'ST', 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)
```

(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 to initialize an instance of `AnalyzeGroup_ST_BUD`.

```
end

a = AnalyzeGroup_ST_BUD('GR', gr, 'DENSITIES', 5:10:35); (6)

gui = GUIElement('PE', a, 'CLOSEREQ', false);(7)
gui.get('DRAW')(8)
gui.get('SHOW')(9)

gui.get('CLOSE')(10)

%%% itest!(11)
%%%% iname!
GUI - Comparison
%%%% iprobability!
.01
%%%% icode!
im_ba = ImporterBrainAtlasXLS('FILE', 'destrieux_atlas.xlsx');
ba = im_ba.get('BA');

gr1 = Group('SUB_CLASS', 'SubjectST', 'SUB_DICT', IndexedDictionary('
     IT_CLASS', 'SubjectST'));
for i = 1:1:50
    sub = SubjectST( ...
        'ID', ['SUB ST ' int2str(i)], ...
        'LABEL', ['Subject ST ' int2str(i)], ...
        'NOTES', ['Notes on subject ST ' int2str(i)], ...
        'BA', ba, ...
        'ST', 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)))
    gr1.get('SUB_DICT').get('ADD', sub)
end

gr2 = Group('SUB_CLASS', 'SubjectST', 'SUB_DICT', IndexedDictionary('
     IT_CLASS', 'SubjectST'));
for i = 1:1:50
    sub = SubjectST( ...
        'ID', ['SUB ST ' int2str(i)], ...
        'LABEL', ['Subject ST ' int2str(i)], ...
        'NOTES', ['Notes on subject ST ' int2str(i)], ...
        'BA', ba, ...
        'ST', 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)))
    gr2.get('SUB_DICT').get('ADD', sub)
end

a1 = AnalyzeGroup_ST_BUD('GR', gr1, 'DENSITIES', 5:10:35);(12)
a2 = AnalyzeGroup_ST_BUD('GR', gr2, 'TEMPLATE', a1); (13)

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) initializes the first `AnalyzeGroup_ST_BUD` similar to the previous test, using 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 the associated GUI of `AnalyzeGroup_ST_BUD`