Implement a New Ensemble Analysis

The BRAPH 2 Developers

January 4, 2025

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

Contents

Implementation of the ensemble analysis

You will implement in detail `AnalyzeEnsemble_CON_BUD`, a direct extension of `AnalyzeEnsemble`. An `AnalyzeEnsemble_CON_BUD` processes connectivity data to construct binary undirected graphs at fixed densities.

Basic properties

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

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

```
1  %% iheader!
2  AnalyzeEnsemble_CON_BUD < AnalyzeEnsemble (a, graph analysis with
        connectivity data of fixed density) is an ensemble-based graph analysis
        using connectivity data of fixed density. ①
3
4  %%% idescription! ②
5  This ensemble-based graph analysis (AnalyzeEnsemble_CON_BUD) analyzes
6  connectivity data using binary undirected multigraphs at fixed densities.
7
8  %%% iseealso!
9  SubjectCON, MultigraphBUD
10
11 %%% ibuild! ③
12 1
```

① defines `AnalyzeEnsemble_CON_BUD` as a subclass of `AnalyzeEnsemble`. The moniker will be a.
② provides a description of this ensemble analysis.

③ defines the build number of the ensemble analysis element.

Code 2: **Basic properties of AnalyzeEnsemble_CON_BUD.** This section of the generator code in `_AnalyzeEnsemble_CON_BUD.gen.m` updates the basic properties required to describe the `AnalyzeEnsemble_CON_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 ensemble-based graph analysis
        using connectivity data of fixed density.
5  %%%% idefault!
6  'AnalyzeEnsemble_CON_BUD'
7
8  %%% iprop!
9  NAME (constant, string) is the name of the ensemble-based graph analysis
        using connectivity data of fixed density.
10 %%%% idefault!
11 'Connectivity Binary Undirected at fixed Density Analyze Ensemble'
12
13 %%% iprop!
```

```
14  DESCRIPTION (constant, string) is the description of the ensemble-based
        graph analysis using connectivity data of fixed density.
15  %%%% idefault!
16  'This ensemble-based graph analysis (AnalyzeEnsemble_CON_BUD) analyzes
        connectivity data using binary undirected multigraphs at fixed
        densities.'
17
18  %%% iprop!
19  TEMPLATE (parameter, item) is the template of the ensemble-based graph
        analysis using connectivity data of fixed density.
20  %%%% isettings!
21  'AnalyzeEnsemble_CON_BUD'
22
23  %%% iprop!
24  ID (data, string) is a few-letter code for the ensemble-based graph analysis
         using connectivity data of fixed density.
25  %%%% idefault!
26  'AnalyzeEnsemble_CON_BUD ID'
27
28  %%% iprop!
29  LABEL (metadata, string) is an extended label of the ensemble-based graph
        analysis using connectivity data of fixed density.
30  %%%% idefault!
31  'AnalyzeEnsemble_CON_BUD label'
32
33  %%% iprop!
34  NOTES (metadata, string) are some specific notes about the ensemble-based
        graph analysis using connectivity data of fixed density.
35  %%%% idefault!
36  'AnalyzeEnsemble_CON_BUD notes'
```

Functionality-focused properties

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

Code 3: **Implementation properties of AnalyzeEnsemble_CON_BUD.** This section of the generator code in `_AnalyzeEnsemble_CON_BUD.gen.m` updates the properties to be used, including `GR` for defining subjects' data, `GRAPH_TEMPLATE` for specifying graph type and parameters, and `G_DICT` for managing graph instances across subjects.

```
1
2  %%% iprop!  (1)
3  GR (data, item) is the subject group, which also defines the subject class
        SubjectCON.
4  %%%% idefault!
5  Group('SUB_CLASS', 'SubjectCON')
6
7  %%% iprop!  (2)
8  GRAPH_TEMPLATE (parameter, item) is the graph template to set all graph and
        measure parameters.
9  %%%% isettings!
10  'MultigraphBUD'
```

(1) defines the property `GR`, which stores the subjects using `SubjectCON` element, containing the subjects' data to be analyzed.

(2) Specifies the `GRAPH_TEMPLATE` to define parameters such as `DENSITIES`, `SEMIPOSITIVIZE_RULE`, and `STANDARDIZE_RULE`. These settings are applied to all graphs in (3). Here, the graph element used is `MultigraphBUD`.

```
11
12  %%% iprop!  ③
13  G_DICT (result, idict) is the graph (MultigraphBUD) ensemble obtained from
        this analysis.
14  %%%% isettings!
15  'MultigraphBUD'
16  %%%% icalculate!
17  g_dict = IndexedDictionary('IT_CLASS', 'MultigraphBUD');
18  gr = a.get('GR');
19  densities = a.get('DENSITIES');  ④
20
21  for i = 1:1:gr.get('SUB_DICT').get('LENGTH')  ⑤
22    sub = gr.get('SUB_DICT').get('IT', i);
23      g = MultigraphBUD( ...⑥
24          'ID', ['graph ' sub.get('ID')], ...
25          'B', sub.getCallback('CON'), ...
26          'DENSITIES', densities, ...  ⑦
27          'LAYERLABELS', cellfun(@(x) [num2str(x) '%'], num2cell(densities), '
        UniformOutput', false), ...
28          'NODELABELS', a.get('GR').get('SUB_DICT').get('IT', 1).get('BA').get
        ('BR_DICT').get('KEYS') ...
29          );
30      g_dict.get('ADD', g)  ⑧
31  end
32
33  if ~isa(a.get('GRAPH_TEMPLATE'), 'NoValue')
34      for i = 1:1:g_dict.get('LENGTH')
35          g_dict.get('IT', i).set('TEMPLATE', a.get('GRAPH_TEMPLATE'))  ⑨
36      end
37  end
38
39  value = g_dict;
40
41  %%% iprop!
42  ME_DICT (result, idict) contains the calculated measures of the graph
        ensemble.
```

③ creates G_DICT, a graph dictionary that contains instances of MultigraphBUD. These instances are derived from the subjects defined in ①.

④ retrieves the densities defined in the new properties below, which is used to configure the MultigraphBUD instances for the analysis.

⑤, ⑥, ⑦, and ⑧ collectively build the graph dictionary (G_DICT). This process begins by iterating over each subject in GR, constructing an instance of MultigraphBUD for each subject based on their respective data, applying the specified DENSITIES parameter, and finally adding the created MultigraphBUD instances into the dictionary.

⑨ ensures that all MultigraphBUD instances in the dictionary are updated with the pre-defined parameters from the graph template specified in ②, if explicitly set by the user during initialization of AnalyzeEnsemble_CON_BUD.

Code 4: **AnalyzeEnsemble_CON_BUD element props.** The props section of the generator code in _AnalyzeEnsemble_CON_BUD.gen.m defines the properties to be used in AnalyzeEnsemble_CON_BUD.

```
1  %% iprops!
2
3  %%% iprop!  ①
4  DENSITIES (parameter, rvector) is the vector of densities.
5  %%%% idefault!
6  [1:1:10]
7  %%%% igui!  ②
8  pr = PanelPropRVectorSmart('EL', a, 'PROP', AnalyzeEnsemble_CON_BUD.
        DENSITIES, ...
9      'MIN', 0, 'MAX', 100, ...
10     'DEFAULT', AnalyzeEnsemble_CON_BUD.getPropDefault('DENSITIES'), ...
11     varargin{:});
12  %%%% ipostset!  ③
13  a.memorize('GRAPH_TEMPLATE').set('DENSITIES', a.getCallback('DENSITIES'));
```

① defines the densities for binarizing the connectivity matrix in a binary undirected multigraph, used in ⑦ of Code 3

② PanelPropRVectorSmart plots a GUI row vector panel for defining densities, supporting MATLAB expressions and limiting values between MIN and MAX.

③ handles postprocessing after DENSITIES is set, memorizing a GRAPH_TEMPLATE with the defined DENSITIES, applied later in ⑨ of Code 3.

Verification through testing

This section validates AnalyzeEnsemble_CON_BUD by implementing tests to confirm its functionality via example scripts and ensure GUI integration.

Code 5: **AnalyzeEnsemble_CON_BUD element tests.** The tests section in the element generator _AnalyzeEnsemble_CON_BUD.gen.m includes logic tests, which verify 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  %%% iexcluded_props!    ①
5  [AnalyzeEnsemble_CON_BUD.TEMPLATE AnalyzeEnsemble_CON_BUD.GRAPH_TEMPLATE]
6
7  %%% itest!  ②
8  %%%% iname!
9  Example
10 %%%% iprobability!    ③
11 .01
12 %%%% icode!
13 create_data_CON_XLS() % only creates files if the example folder doesn't
       already exist
14
15 example_CON_BUD
16
17 %%% itest!  ④
18 %%%% iname!
19 GUI - Analysis
20 %%%% iprobability!
21 .01
22 %%%% icode!
23 im_ba = ImporterBrainAtlasXLS('FILE', 'desikan_atlas.xlsx');
24 ba = im_ba.get('BA');  ⑤
25
26 gr = Group('SUB_CLASS', 'SubjectCON', 'SUB_DICT', IndexedDictionary('
       IT_CLASS', 'SubjectCON'));  ⑥
27 for i = 1:1:50
28     sub = SubjectCON( ...
29         'ID', ['SUB CON ' int2str(i)], ...
30         'LABEL', ['Subject CON ' int2str(i)], ...
31         'NOTES', ['Notes on subject CON ' int2str(i)], ...
32         'BA', ba, ...
33         'CON', rand(ba.get('BR_DICT').get('LENGTH')) ...
34         );
35     sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
      rand()))
36     sub.memorize('VOI_DICT').get('ADD', VOICategoric('ID', 'Sex', '
      CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
37     gr.get('SUB_DICT').get('ADD', sub)
38 end
39
40 a = AnalyzeEnsemble_CON_BUD('GR', gr, 'DENSITIES', 5:5:20);  ⑦
41
```

① List of properties that are excluded from testing.

② Tests the functionality of AnalyzeEnsemble_CON_BUD using an example script.

③ assigns a low test execution probability.

④ Tests the direct GUI functionality of AnalyzeEnsemble_CON_BUD.

⑤ and ⑥ define the necessary objects required to initialize an instance of AnalyzeEnsemble_CON_BUD.

⑦ initializes an AnalyzeEnsemble_CON_BUD instance using the specified gr (group) and densities.

```
42  gui = GUIElement('PE', a, 'CLOSEREQ', false); (8)
43  gui.get('DRAW') (9)
44  gui.get('SHOW') (10)
45
46  gui.get('CLOSE') (11)
47
48  %%% ¡test! (12)
49  %%%% ¡name!
50  GUI - Comparison
51  %%%% ¡probability!
52  .01
53  %%%% ¡code!
54  im_ba = ImporterBrainAtlasXLS('FILE', 'desikan_atlas.xlsx');
55  ba = im_ba.get('BA');
56
57  gr1 = Group('SUB_CLASS', 'SubjectCON', 'SUB_DICT', IndexedDictionary('
         IT_CLASS', 'SubjectCON'));
58  for i = 1:1:50
59      sub = SubjectCON( ...
60          'ID', ['SUB CON ' int2str(i)], ...
61          'LABEL', ['Subject CON ' int2str(i)], ...
62          'NOTES', ['Notes on subject CON ' int2str(i)], ...
63          'BA', ba, ...
64          'CON', rand(ba.get('BR_DICT').get('LENGTH')) ...
65          );
66      sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
          rand()))
67      sub.memorize('VOI_DICT').get('ADD', VOICategoric('ID', 'Sex', '
         CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
68      gr1.get('SUB_DICT').get('ADD', sub)
69  end
70
71  gr2 = Group('SUB_CLASS', 'SubjectCON', 'SUB_DICT', IndexedDictionary('
         IT_CLASS', 'SubjectCON'));
72  for i = 1:1:50
73      sub = SubjectCON( ...
74          'ID', ['SUB CON ' int2str(i)], ...
75          'LABEL', ['Subject CON ' int2str(i)], ...
76          'NOTES', ['Notes on subject CON ' int2str(i)], ...
77          'BA', ba, ...
78          'CON', rand(ba.get('BR_DICT').get('LENGTH')) ...
79          );
80      sub.memorize('VOI_DICT').get('ADD', VOINumeric('ID', 'Age', 'V', 100 *
          rand()))
81      sub.memorize('VOI_DICT').get('ADD', VOICategoric('ID', 'Sex', '
         CATEGORIES', {'Female', 'Male'}, 'V', randi(2, 1)))
82      gr2.get('SUB_DICT').get('ADD', sub)
83  end
84
85  a1 = AnalyzeEnsemble_CON_BUD('GR', gr1, 'DENSITIES', 5:5:20); (13)
86  a2 = AnalyzeEnsemble\_CON\_BUD('GR', gr2, 'TEMPLATE', a1); (14)
87
88  c = CompareEnsemble( ... (15)
89      'P', 10, ...
90      'A1', a1, ...
91      'A2', a2, ...
92      'WAITBAR', true, ...
```

(8), (9), and (10) test the process of creating a GUI for AnalyzeEnsemble_CON_BUD, drawing it, and showing it on the screen.

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

(12) tests the associated GUI functionality of AnalyzeEnsemble_CON_BUD.

(13) Similar to the previous test, this initializes the first AnalyzeEnsemble_CON_BUD with the specified gr and densities.

(14) initializes the second AnalyzeEnsemble_CON_BUD using the first AnalyzeEnsemble_CON_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.

(15) creates a CompareEnsemble instance with the defined AnalyzeEnsemble_CON_BUD instances.

```
93      'VERBOSE', false, ...
94      'MEMORIZE', true ...
95      );
96
97   gui = GUIElement('PE', c, 'CLOSEREQ', false);    (16)
98   gui.get('DRAW')   (17)
99   gui.get('SHOW')   (18)
100
101  gui.get('CLOSE')   (19)
```

(16), (17), (18), and (19) test creating, drawing, showing, and closing the GUI of the CompareEnsemble, which is an associated GUI of AnalyzeEnsemble_CON_BUD