Implement a new Ensemble Analysis

The BRAPH 2 Developers

December 23, 2024

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 with fixed densities (BUD).

Contents

Implementatoin of the ensemble analysis

You will implement in detail `AnalyzeEnsemble_CON_BUD`, which is a direct extension of `AnalyzeEnsemble`.

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.

```
%% !header!
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)

%%% !description! (2)
This ensemble-based graph analysis (AnalyzeEnsemble_CON_BUD) analyzes
connectivity data using binary undirected multigraphs with fixed densities.

%%% !seealso!
SubjectCON, MultigraphBUD

%%% !build! (3)
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: **AnalyzeEnsemble_CON_BUD element prop update.** The `props_update` section of the generator code in `_AnalyzeEnsemble_CON_BUD.gen.m` updates the properties of the `AnalyzeEnsemble` element. This defines the core properties of the ensemble analysis.

```
%% !props_update!

%%% !prop!
ELCLASS (constant, string) is the class of the ensemble-based graph analysis
      using connectivity data of fixed density.
%%%% !default!
'AnalyzeEnsemble_CON_BUD'

%%% !prop!
NAME (constant, string) is the name of the ensemble-based graph analysis
     using connectivity data of fixed density.
%%%% !default!
'Connectivity Binary Undirected at fixed Density Analyze Ensemble'

%%% !prop!
DESCRIPTION (constant, string) is the description of the ensemble-based
     graph analysis using connectivity data of fixed density.
%%%% !default!
'This ensemble-based graph analysis (AnalyzeEnsemble_CON_BUD) analyzes
     connectivity data using binary undirected multigraphs with fixed
     densities.'

%%% !prop!
TEMPLATE (parameter, item) is the template of the ensemble-based graph
     analysis using connectivity data of fixed density.
%%%% !settings!
```

```
'AnalyzeEnsemble_CON_BUD'

%%% iprop!
ID (data, string) is a few-letter code for the ensemble-based graph analysis
      using connectivity data of fixed density.
%%%% idefault!
'AnalyzeEnsemble_CON_BUD ID'

%%% iprop!
LABEL (metadata, string) is an extended label of the ensemble-based graph
     analysis using connectivity data of fixed density.
%%%% idefault!
'AnalyzeEnsemble_CON_BUD label'

%%% iprop!
NOTES (metadata, string) are some specific notes about the ensemble-based
     graph analysis using connectivity data of fixed density.
%%%% idefault!
'AnalyzeEnsemble_CON_BUD notes'

%%% iprop! (1)
GR (data, item) is the subject group, which also defines the subject class
     SubjectCON.
%%%% idefault!
Group('SUB_CLASS', 'SubjectCON')

%%% iprop! (2)
GRAPH_TEMPLATE (parameter, item) is the graph template to set all graph and
     measure parameters.
%%%% isettings!
'MultigraphBUD'

%%% iprop! (3)
G_DICT (result, idict) is the graph (MultigraphBUD) ensemble obtained from
     this analysis.
%%%% isettings!
'MultigraphBUD'
%%%% icalculate!
g_dict = IndexedDictionary('IT_CLASS', 'MultigraphBUD');
gr = a.get('GR');
densities = a.get('DENSITIES'); (4)

for i = 1:1:gr.get('SUB_DICT').get('LENGTH') (5)
  sub = gr.get('SUB_DICT').get('IT', i);
    g = MultigraphBUD( ...(7)
        'ID', ['graph ' sub.get('ID')], ...
        'B', sub.getCallback('CON'), ...
        'DENSITIES', densities, ... (8)
        'LAYERLABELS', cellfun(@(x) [num2str(x) '%'], num2cell(densities), '
     UniformOutput', false), ...
        'NODELABELS', a.get('GR').get('SUB_DICT').get('IT', 1).get('BA').get
     ('BR_DICT').get('KEYS') ...
        );
    g_dict.get('ADD', g) (9)
end

if ~isa(a.get('GRAPH_TEMPLATE'), 'NoValue')
    for i = 1:1:g_dict.get('LENGTH')
        g_dict.get('IT', i).set('TEMPLATE', a.get('GRAPH_TEMPLATE'))  (11)
```

(1) defines the GR which contains the subjects data with SubjectCON element to be analyzed on.

(2) defines GRAPH_TEMPLATE the graph template with specified parameters, such as DENSITIES, SEMIPOSITIVIZE_RULE, and STANDARDIZE_RULE, which will be used to set up for all graphs in (4). In this example, the graph element is MultigraphBUD.

(3) creates G_DICT a graph dictionary which contains all of MultigraphBUD derived respectively from (1).

(4) retrieves the densities for setting up MultigraphBUD, defined in the new properties below.

(5), (6), (7), and (8) create the dictionary of graph. It starts by looping through every subject in GR, creating MultigraphBUD based on each subject data. Then it sets up DENSTITIES the parameter of the MultigraphBUD, and finally add the MultigraphBUD into the dictionary.

(11) sets all the MultigraphBUD in the dictionary with all the specified parameteres definded in (2) if the graph template is given by a user.

```
72      end
73  end
74
75  value = g_dict;
76
77  %%% iprop!
78  ME_DICT (result, idict) contains the calculated measures of the graph
        ensemble.
```