

# Package ‘discourseGT’

April 2, 2021

**Type** Package

**Title** Analyze Group Patterns using Graph Theory in Educational Settings

**Version** 1.1.5

**Date** 2021-4-2

**Description** Analyzes group patterns using discourse analysis data with graph theory mathematics. Takes the order of which individuals talk and converts it to a network edge and weight list. Returns the density, centrality, centralization, and subgroup information for each group. Based on the analytical framework laid out in Chai et al. (2019) <doi:10.1187/cbe.18-11-0222>.

**License** MIT + file LICENSE

**LazyData** true

**Encoding** UTF-8

**Depends** R (>= 3.5.0)

**Suggests**

**RoxygenNote** 7.0.2

**Imports** igraph, ggpubr, GGally, network, ggplot2, dplyr, ggrepel, BiocManager, sna

**NeedsCompilation** no

**Author** Albert Chai [aut],  
Andrew Lee [aut],  
Joshua Le [aut, cre],  
Katherine Ly [ctb],  
Kevin Banh [ctb],  
Priya Pahal [ctb],  
Stanley Lo [aut]

**Maintainer** Joshua Le <jpl038@ucsd.edu>

**Repository** CRAN

**Date/Publication** 2021-04-02 19:00:02 UTC

## R topics documented:

attributeData . . . . .	2
basicPlot . . . . .	3
coreNetAnalysis . . . . .	4
discourseGT . . . . .	4
edgelist_raw . . . . .	5
plot1Att . . . . .	5
plot2Att . . . . .	6
plotNGTData . . . . .	7
prepareGraphs . . . . .	8
sampleData1 . . . . .	9
subgroupsNetAnalysis . . . . .	9
summaryNet . . . . .	10
tabulate_edges . . . . .	11
writeData . . . . .	12

<b>Index</b>	<b>13</b>
--------------	-----------

---

attributeData	<i>Sample Attribute Data</i>
---------------	------------------------------

---

### Description

A data set that sample attribute data to complement Sample Data 1. This data was randomly generated. This data set is for students who are currently taking a STEM course at a major university. 4 students are in this sample study.

### Usage

```
attributeData
```

### Format

A data frame of 4 rows with 12 variables

**node** Student identifier in the group

**gender** Gender of student

**ethnicity** Ethnicity of student

**current\_gpa** Current overall GPA of student

**first\_generation** If the student is a first generation standing

**stem\_major** If student is in a STEM major

**major** Major of the student

**course\_reason** Reasons to why student is taking the course

**class\_level** Current class standing at the university

**number\_prior\_ap** Number of prior AP courses taken

**residency** Current residency of the student

**sat\_score** SAT score for admission ...

---

basicPlot

*Plot Graphs*

---

### Description

Plots the graph using the base plot function. To map attributes on the graph use 'plot1Att' for 1 attribute or 'plot2Att' for 2 attributes.

### Usage

```
basicPlot(
  ginp,
  graph_selection_input = 0,
  curvedEdgeLines = TRUE,
  arrowSizeMultiplier = 1,
  scaledEdgeLines = FALSE,
  scaledMin = NULL,
  scaledMax = NULL
)
```

### Arguments

**ginp** The prepared graph object from prepareGraphs function

**graph\_selection\_input** The type of graphical projection to be used. Default projection is 0 (Fruchterman Reingold). Selection must be a numeric option from 0-2. Other options include: 1 = Kamada Kawai, 2 = Reingold Tilford

**curvedEdgeLines** Whether or not the edges between nodes should be curved or straight. Default is curved lines.

**arrowSizeMultiplier** Adjusts the default arrow size based on a multiplier. Default value is 1.

**scaledEdgeLines** Whether or not the edges of the graph should be scaled

**scaledMin** If scaledEdgeLines = TRUE, then what the lightest weight should be scaled to

**scaledMax** If scaledEdgeLines = TRUE, then what the heaviest weight should be scaled to

### Value

Returns graphical plot to disk, if selected, or to R console

**Examples**

```
df <- sampleData1
prepNet <- tabulate_edges(df, iscsvfile = FALSE, silentNodes = 0)
baseNet <- prepareGraphs(prepNet, project_title = "Sample Data 1", weightedGraph = TRUE)

#Plot the graph
basicPlot(baseNet)
```

---

 coreNetAnalysis

*Run Graphical Analysis Core Parameters*


---

**Description**

Analyzes the graphs with the core parameters, such as number of edges and nodes, density, average degree, centrality, and modularity

**Usage**

```
coreNetAnalysis(ginp)
```

**Arguments**

ginp                    The prepared graph object from prepareGraphs function

**Value**

Gives the edge and weighted edge counts, number of nodes, density, degree (averages), memberships, modularity, centrality, articulation points, and strong/weak plots as a list object

**Examples**

```
df <- sampleData1
prepNet <- tabulate_edges(df, iscsvfile = FALSE, silentNodes = 0)
baseNet <- prepareGraphs(prepNet, project_title = "Sample Data 1", weightedGraph = TRUE)

coreNetAnalysis(baseNet)
```

---

 discourseGT

*discourseGT: A package to analyze groups patterns using graph theory in educational settings*


---

**Description**

discourseGT is a library that utilizes the various network libraries to produce network analysis in educational settings

---

edgelist_raw	<i>Process raw order lists from two column format to edge lists</i>
--------------	---

---

**Description**

Takes raw input that is in a 2 column format/question-and-response format and generates an appropriate edge lists in a combined .csv file.

**Usage**

```
edgelist_raw(input_file, iscsvfile = TRUE)
```

**Arguments**

input_file	Source of the raw input file. Must be in a .csv format
iscsvfile	Sets if the input_file is a csv file or a R data frame object

**Value**

Saves the weight and edge lists as a data.frame object or a .csv file to disk.

**Examples**

```
df <- sampleData1  
prepNet <- edgelist_raw(df, iscsvfile = FALSE)
```

---

plot1Att	<i>Plots Graphs using ggplot2 with one attribute</i>
----------	--

---

**Description**

Plots graph data using the GGally library and ggnet function while incorporating demographic properties. Use this plot function if you have all demographic data available to plot.

**Usage**

```
plot1Att(  
  data,  
  prop = 20,  
  graphmode = "fruchtermanreingold",  
  attribute = NULL,  
  attribute.label = NULL,  
  attribute.node.labels = NULL,  
  attribute.nodesize = 10  
)
```

**Arguments**

<code>data</code>	Data from the <code>prepareGraphs</code> function
<code>prop</code>	Rescaling the graph edge sizes for the plot
<code>graphmode</code>	Type of graphical projection to use. Default is Fruchterman Reingold. Refer to <code>gplot.layout</code> for the various available options
<code>attribute</code>	Mapping to the attribute information, can be list or column in data frame
<code>attribute.label</code>	Name of the attribute info (Required)
<code>attribute.node.labels</code>	Mapping to the node labels, can be list or column in data frame
<code>attribute.nodesize</code>	Size of the nodes. Default will result in size of 10. Can be replaced with custom mapping in list or column in data frame. (Required)

**Examples**

```
df <- sampleData1
prepNet <- tabulate_edges(df, iscsvfile = FALSE, silentNodes = 0)
baseNet <- prepareGraphs(prepNet, project_title = "Sample Data 1", weightedGraph = TRUE)
attdata <- attributeData
plot1Att(baseNet, prop = 20, graphmode = "fruchtermanreingold",
attribute = attdata$gender,
attribute.label = "Gender",
attribute.node.labels = attdata$node, attribute.nodesize = 12)
```

---

plot2Att

*Plots Graphs using ggplot2 with two attributes*


---

**Description**

Plots graph data using the GGally library and `ggnet` function while incorporating demographic properties. Use this plot function if you have all demographic data available to plot.

**Usage**

```
plot2Att(
  data,
  prop = 20,
  graphmode = "fruchtermanreingold",
  attribute1 = NULL,
  attribute2 = NULL,
  attribute1.label = "Attribute 1",
  attribute2.label = "Attribute 2",
  attribute.node.labels = NULL,
  attribute.nodesize = 10
)
```

**Arguments**

data	Data from the prepareGraphs function
prop	Rescaling the graph edge sizes for the plot
graphmode	Type of graphical projection to use. Default is Fruchterman Reingold. Refer to gplot.layout for the various available options
attribute1	Mapping to the attribute 1 information, can be list or column in data frame (Required)
attribute2	Mapping to the attribute 2 information, can be list or column in data frame (Required)
attribute1.label	Name of the attribute 1 info (Required)
attribute2.label	Name of the attribute 2 info (Required)
attribute.node.labels	Mapping to the node labels, can be list or column in data frame (Required)
attribute.nodesize	Size of the nodes. Default will result in size of 10. Can be replaced with custom mapping in list or column in data frame. (Required)

**Examples**

```
df <- sampleData1
prepNet <- tabulate_edges(df, iscsvfile = FALSE, silentNodes = 0)
baseNet <- prepareGraphs(prepNet, project_title = "Sample Data 1", weightedGraph = TRUE)
attdata <- attributeData
plot2Att(baseNet, prop = 20, graphmode = "fruchtermanreingold",
attribute1 = attdata$gender, attribute2 = attdata$ethnicity,
attribute1.label = "Gender", attribute2.label = "Ethnicity",
attribute.node.labels = attdata$node, attribute.nodesize = 12)
```

---

plotNGTData

*Plot non-graphical parameters*


---

**Description**

Creates plots for non-graph theory parameters for episode lengths, questions per hour versus responses per hour, and normalized turn ratio

**Usage**

```
plotNGTData(data, convoMinutes, iscsvfile = TRUE, silentNodes = 0)
```

**Arguments**

data	Original raw input data in ordered question/response 2 column format
convoMinutes	Time length of the conversation in the graph in minutes
iscsvfile	Sets if the input data file to function is a .csv file or a R data frame object
silentNodes	The number of nodes that do not interact with other nodes but are in the group

**Value**

Creates a plot returning the questions per hour versus responses per hour, frequency plot of the number of episodes, and normalized turn ratio

**Examples**

```
df <- sampleData1
plotNGTData(df, convoMinutes = 60, iscsvfile = FALSE, silentNodes = 0)
```

---

prepareGraphs	<i>Prepare Graphs</i>
---------------	-----------------------

---

**Description**

Prepares the graphical object from the prepared edge and weight list data frame

**Usage**

```
prepareGraphs(raw_data_input, project_title = "", weightedGraph = TRUE)
```

**Arguments**

raw_data_input	The raw edge and weight list processed from the tabulate_edges() function.
project_title	The title of the project.
weightedGraph	Graph will add weights to the edges to a set of nodes based on the weight specified on the list. Default allows for weights on the graph.

**Value**

Stores the igraph graph object, graph adjacency matrix, edge and weight lists, project title, and a user option for weighted to list object.

**Examples**

```
df <- sampleData1
prepNet <- tabulate_edges(df, iscsvfile = FALSE, silentNodes = 0)
baseNet <- prepareGraphs(prepNet, project_title = "Sample Data 1", weightedGraph = TRUE)
```



---

 sampleData1

*Sample Episode Start and Episode Continuation Data (2 Column)*


---

### Description

A data set that contains Episode Start (in this case, questions) and Episode Continuation (in this case, responses) data in a 2 column format. This data was randomly generated. Case basis: 4 students in a group discussion questions in a STEM course at a major university.

### Usage

```
sampleData1
```

### Format

A data frame with 466 rows and 2 variables:

**ep\_start** If participant in the graph has initiated an episode

**ep\_cont** If participant in the graph has raised a response to the previous episode ...

---

 subgroupsNetAnalysis *Runs subgroup analysis on graphs*


---

### Description

Performs a subgroup analysis on the graph

### Usage

```
subgroupsNetAnalysis(ginp, raw_input = NULL, normalized = FALSE)
```

### Arguments

ginp	The prepared graph object from prepareGraphs function
raw_input	The data of the original .csv file
normalized	Normalize the betweenness centrality values

### Value

Saves number of potential cliques, cores, symmetry of the graph, dyads in graphs, node composition in proposed cliques, neighbors adjacent to each node, transitivity (local and global) as a list object

**Examples**

```
df <- sampleData1
prepNet <- tabulate_edges(df, iscsvfile = FALSE, silentNodes = 0)
baseNet <- prepareGraphs(prepNet, project_title = "Sample Data 1", weightedGraph = TRUE)
subgroupsNetAnalysis(baseNet, raw_input = df)
```

---

summaryNet	<i>Print summary of graph results</i>
------------	---------------------------------------

---

**Description**

Returns a summary of the processed graph results on console. The initial graph configuration and core analysis is required for this function to work. The other components are optional due to the modular nature of the functions. Data must be stored as a data object.

**Usage**

```
summaryNet(
  netintconfigData = NULL,
  coreNetAnalysisData = NULL,
  subgroupsNetAnalysisData = NULL,
  display = FALSE
)
```

**Arguments**

netintconfigData	Data object where the graph configuration data is stored (from 'prepareGraphs')
coreNetAnalysisData	Data object where the core analysis data is stored (from 'coreNetAnalysis')
subgroupsNetAnalysisData	Data object where subgroup analysis data is stored (from 'subgroupsNetAnalysis')
display	Should the output be displayed in the R console? Results are saved as the project name in the initial config data as a text file on disk.

**Value**

Prints organized summary of all results of the graph with modular components on console or to .txt file on disk.

**Examples**

```
df <- sampleData1
prepNet <- tabulate_edges(df, iscsvfile = FALSE, silentNodes = 0)
prepGraphs <- prepareGraphs(prepNet, project_title = "Sample Data 1", weightedGraph = TRUE)
coreNet <- coreNetAnalysis(prepGraphs)
subgroup <- subgroupsNetAnalysis(prepGraphs, raw_input = df)
summaryNet(netintconfigData = prepGraphs, coreNetAnalysisData = coreNet,
subgroupsNetAnalysisData = subgroup, display = TRUE)
```

---

tabulate_edges	<i>Process raw order lists from two column format to edge and weight lists</i>
----------------	--

---

**Description**

Takes raw input that is in a 2 column format/question-and-response format and generates an appropriate edge and weight lists in a combined .csv file. The weights in this function are determined by the number of occurrences a specific edge has occurred in the graph

**Usage**

```
tabulate_edges(input, iscsvfile = TRUE, silentNodes = 0)
```

**Arguments**

input	Input in question-and-response format. Must be a data.frame or file name of a .csv
iscsvfile	Sets if the input is a csv file or a R data frame object
silentNodes	The number of nodes that do not interact with other nodes but are in the group

**Value**

Saves the weight and edge lists as a data.frame object or a .csv file to disk.

**Examples**

```
df <- sampleData1
tabData <- tabulate_edges(df, iscsvfile = FALSE, silentNodes = 0)
```

---

`writeData`*Exports graphs and data objects from the package to disk*

---

**Description**

Saves information from graphs and data objects created by package. Plots are saved as .tiff at 300 dpi

**Usage**

```
writeData(project_name, objectfile, dirpath = NULL)
```

**Arguments**

<code>project_name</code>	Name of the project
<code>objectfile</code>	The saved data object data file
<code>dirpath</code>	The working directory that the files will be saved to. Path required for write function to work. Current directory, use "." as the dirpath

**Value**

Saves the requested object file to disk. Saves graphs or summary information sheets.

**Examples**

```
attributeData <- attributeData
df <- sampleData1

prepNet <- tabulate_edges(df, iscsvfile = FALSE, silentNodes = 0)
baseNet <- prepareGraphs(prepNet, project_title = "Sample Data 1", weightedGraph = TRUE)
NetPlots2 <- plot2Att(baseNet, attribute1 = attributeData$ethnicity,
  attribute2 = attributeData$gender, attribute.node.labels = attributeData$node,
  attribute1.label = "Ethnicity", attribute2.label = "Gender")

writeData("Sample Data 1", NetPlots2, dirpath = tempdir())
```

# Index

## \* datasets

- attributeData, 2
- sampleData1, 9
  
- attributeData, 2
  
- basicPlot, 3
  
- coreNetAnalysis, 4
  
- discourseGT, 4
  
- edgelist\_raw, 5
  
- plot1Att, 5
- plot2Att, 6
- plotNGTData, 7
- prepareGraphs, 8
  
- sampleData1, 9
- subgroupsNetAnalysis, 9
- summaryNet, 10
  
- tabulate\_edges, 11
  
- writeData, 12