Package 'grandpa'

December 20, 2022

Type Package

Title GeneRAtive Networks using Property and Degree Augmentation

Version 0.1.0

Author Carly Bobak, Yifan Zhao

Description This packages generates graphs from existing graphs using both observed attribute structure and topological augmentation

License MIT

Depends R (¿= 3.5.0)

Encoding UTF-8

LazyData true

Imports igraph,
tidyr,
dplyr (¿= 1.0.0),
ggplot2

RoxygenNote 7.2.2

R topics documented:

---

addAugment *Add degree augmentation label*

Description

This function is typically called by the grandpa function rather than directly.

Usage

```
addAugment(G, nbins, degType)
```

Arguments

G	an igraph object
nbins	the number of degree bins used in the degree augmentation
degType	string; the type of degree to be used in the degree augmentation. See ?degree for details

Value

an igraph object containing an augmentation label corresponding to degree

Examples

```
G <- make_graph("Zachary")
G <- addAugment(G,3,"all")
```

---

addCommunityAugment *Add community augmentation label*

Description

This function is typically called by the grandpa function rather than directly. By default, this function implements louvain clustering to detect community membership. Note, it is recommended to fit a community algorithm outside of the GRANDPA framework and add a vertext label called 'CommunityLabel' for best results.

Usage

```
addCommunityAugment(G)
```

Arguments

G	an igraph object

Value

an igraph object containing an augmentation label corresponding to community membership

Examples

```
G <- make_graph("Zachary")
G <- addCommunityAugment(G)
```

`calcObsProbs` *Calculates the observed edge and vertex probabilities*

Description

This function is typically called by the grandpa function rather than directly.

Usage

```
calcObsProbs(G)
```

Arguments

`G`	An igraph object, containing labels indicated with the word 'Label' that should be used to calculate edge and vertex probabilities

Value

a list of tidy tables containing the vertex probabilities and edge probabilities respectively

Examples

```
G = make_graph("Zachary")
V(G)$degree<-degree(G,mode="all")

V(G)$Label1<-rep("A",vcount(G))
V(G)$Label1[V(G)$degree>median(V(G)$degree)]<-"B"

obsProbTabs<-calcObsProbs(G)
vertexProbs<-obsProbTabs[[1]]
edgeProbs<-obsProbTabs[[2]]
```

`createMap` *Creates the map which lists vertix IDs based on attribute labels*

Description

This function is typically called by the grandpa function rather than directly.

Usage

```
createMap(sampled_v)
```

Arguments

sampled_v	a dataframe containing the sampled vertices and their corresponding labels on each row

Value

a list which contains a mapping of all vertex IDs based on label combinations

Examples

```
G = make_graph("Zachary")
V(G)$degree<-degree(G,mode="all")

V(G)$Label1<-rep("A",vcount(G))
V(G)$Label1[V(G)$degree>median(V(G)$degree)]<-"B"

obsProbTabs<-calcObsProbs(G)
vertexProbs<-obsProbTabs[[1]]
edgeProbs<-obsProbTabs[[2]]

sampledV<-sampleVertices(vertexProbs,50,0)
map<-createMap(sampledV)
```

---

degree_error *Calculates the error between two degree distributions of graphs*

Description

This function takes an original graph and a simulated graph and calculates the normalized root mean square error between their degree distributions.

Usage

```
degree_error(Gs, Gt)
```

Arguments

Gs	the original graph
Gt	the generated graph

Value

numerical; the NRMSE between the degree distributions

grandpa *Conducts a GRANPDA procedure*

Description

Uses the GRANPDA framework to generate a network with matching attribute probabilities

Usage

```
grandpa(
  G,
  nt = vcount(G),
  mt = ecount(G),
  preventSelf = T,
  preventDups = T,
  augment = F,
  augmentCommunity = F,
  nbins = 3,
  degType = "all",
  seed = 0
)
```

Arguments

G	an igraph object containing the original graph. Attributes which will be in the modelling procedure must contain the word 'Label'
nt	the number of desired nodes in the generated graph
mt	the number of desired edges in the generated graph
preventSelf	logical; should self connections be prevented?
preventDups	logical; should duplicate connections be prevented?
augment	logical; should degree augmentation occur?
augmentCommunity	logical; should communited be augmented?
nbins	the number of degree bins used in the degree augmentation
degType	string; the type of degree to be used in the degree augmentation. See ?degree for details
seed	a random seed set for reproducibility

Value

an igraph object with attribute and augmentation labels

Examples

```
G = make_graph("Zachary")
V(G)$degree<-degree(G,mode="all")

V(G)$Label1<-rep("A",vcount(G))
```

```
V(G)$Label1[V(G)$degree>median(V(G)$degree)]<-"B"

GSim<-grandpa(G)
```

`initializeTargetG` *Intializes an empty graph with the generated vertices*

Description

This function is typically called by the grandpa function rather than directly.

Usage

```
initializeTargetG(sampled_v, nt, d)
```

Arguments

`sampled_v`	a dataframe containing the sampled vertices and their corresponding labels on each row
`nt`	the number of desired nodes in the generated graph
`d`	logical; is the graph directed?

Value

an empty igraph object with the desired vertices

Examples

```
G = make_graph("Zachary")
V(G)$degree<-degree(G,mode="all")

V(G)$Label1<-rep("A",vcount(G))
V(G)$Label1[V(G)$degree>median(V(G)$degree)]<-"B"

obsProbTabs<-calcObsProbs(G)
vertexProbs<-obsProbTabs[[1]]
edgeProbs<-obsProbTabs[[2]]

sampledV<-sampleVertices(vertexProbs,50,0)
initG<-initializeTargetG(sampledV,50,FALSE)
```

physNet *Simulated physician network data*

Description

Graph simulated from medicare claims wherein physicians are nodes, and edges between nodes indicate overlapping patients. Physician specialty is represented as a vertex attribute and shown using vertex color by default.

Usage

```
data(physNet)
```

Format

An object of class `"igraph"`; see `igraph`.

Source

arXiv

References

Bobak et al. (2022) arXiv preprint arXiv:2211.15000. (arXiv)

Examples

```
data(physNet)
plot(physNet,vertex.label=NA)
```

preventDuplicateConnections *Prevents duplicate connections during graph generation*

Description

This function is typically called by the grandpa function and should not be used directly.

Usage

```
preventDuplicateConnections(v1, v2, lab1, lab2, C2V, Gt)
```

Arguments

v1	a vertex id
v2	a second vertex id
lab1	the label corresponding to vertex id 1
lab2	the label corresponding to vertex id 2
C2V	a mapping of label categories to vertex IDs
Gt	a graph being generated

Value

updated vertex IDs

`preventSelfConnections` *Prevents self connections during graph generation*

Description

This function is typically called by the grandpa function and should not be used directly.

Usage

```
preventSelfConnections(v1, v2, lab1, lab2, C2V)
```

Arguments

`v1`	a vertex id
`v2`	a second vertex id
`lab1`	the label corresponding to vertex id 1
`lab2`	the label corresponding to vertex id 2
`C2V`	a mapping of label categories to vertex IDs

Value

updated vertex IDs

`sampleEdges` *Creates the edges that will be used in the generated graph*

Description

This function is typically called by the grandpa function rather than directly.

Usage

```
sampleEdges(observedEdgeLabels, mt, seed)
```

Arguments

`observedEdgeLabels`	a dataframe containing the observed edge label combinations and their probabilities from the original graph
`mt`	the number of desired edges in the generated graph
`seed`	the random seed to be used for reproducibility

Value

a dataframe containing edges and their nodel attributes on either end

Examples

```
G = make_graph("Zachary")
V(G)$degree<-degree(G,mode="all")

V(G)$Label1<-rep("A",vcount(G))
V(G)$Label1[V(G)$degree>median(V(G)$degree)]<-"B"

obsProbTabs<-calcObsProbs(G)
vertexProbs<-obsProbTabs[[1]]
edgeProbs<-obsProbTabs[[2]]

sampledE<-sampleVertices(edgeProbs,75,0)
```

---

sampleVertices *Creates the vertices that will be used in the generated graph*

Description

This function is typically called by the grandpa function rather than directly.

Usage

```
sampleVertices(observedLabels, nt, seed)
```

Arguments

observedLabels	a dataframe containing the observed label combinations and their probabilities from the original graph
nt	the number of desired nodes in the generated graph
seed	the random seed to be used for reproducibility

Value

a dataframe containing node attributes

Examples

```
G = make_graph("Zachary")
V(G)$degree<-degree(G,mode="all")

V(G)$Label1<-rep("A",vcount(G))
V(G)$Label1[V(G)$degree>median(V(G)$degree)]<-"B"

obsProbTabs<-calcObsProbs(G)
vertexProbs<-obsProbTabs[[1]]
edgeProbs<-obsProbTabs[[2]]

sampledV<-sampleVertices(vertexProbs,50,0)
```

simGraph *Procedure to add edges to the generated graph*

Description

This function is typically called by the grandpa function rather than directly.

Usage

```
simGraph(Gt, sampled_e, C2V, seed, preventSelf, preventDups)
```

Arguments

Gt	The empty graph
sampled_e	a dataframe containing the edges for the generated graph and their node attributes
C2V	a mapping of the nodal attribute combinations and vertex IDs
seed	the random seed to be used for reproducibility
preventSelf	logical; should self connections be prevented?
preventDups	logical; should duplicate connections be prevented

Value

a generated igraph object

Examples

```
G = make_graph("Zachary")
V(G)$degree<-degree(G,mode="all")

V(G)$Label1<-rep("A",vcount(G))
V(G)$Label1[V(G)$degree>median(V(G)$degree)]<-"B"

obsProbTabs<-calcObsProbs(G)
vertexProbs<-obsProbTabs[[1]]
edgeProbs<-obsProbTabs[[2]]

sampledV<-sampleVertices(vertexProbs,50,0)
initG<-initializeTargetG(sampledV,50,FALSE)
map<-createMap(sampledV)
sampledE<-sampleVertices(edgeProbs,75,0)

GSim<-simGraph(initG,sampledE,map,0,T,T)
```

Index