

# Package ‘QuACN’

February 14, 2012

**Type** Package

**Title** QuACN: Quantitative Analysis of Complex Networks

**Version** 1.3.4

**Date** 2010-05-27

**Author** Laurin Mueller, Michael Schutte, Karl Kugler, Matthias Dehmer

**Maintainer** Laurin Mueller <laurin@eigenlab.net>

**Description** Quantitative Analysis of Complex Networks. This package offers a set of topological network measures to analyze complex Networks structurally.

**License** LGPL

**LazyLoad** yes

**Depends** graph, RBGL, igraph (>= 0.5.5-1), combinat, Rmpfr

**Repository** CRAN

**Date/Publication** 2011-10-18 10:49:55

## R topics documented:

QuACN-package	3
adjacencyMatrix	6
atomBondConnectivity	7
augmentedMatrix	8
augmentedZagreb	9
balabanID	10
balabanJ	11
balabanlike1	12
balabanlike2	13
bertz	14
bonchev1	15
bonchev2	16

bondOrderID . . . . .	17
calculateDescriptors . . . . .	18
compactness . . . . .	21
complexityIndexB . . . . .	22
connectivityID . . . . .	23
degreeDistribution . . . . .	24
diameter . . . . .	25
distanceCodeCentric . . . . .	26
distanceDegreeCentric . . . . .	27
distanceMatrix . . . . .	28
distancePathMatrix . . . . .	28
distSumConnectMatrix . . . . .	29
dobrynin . . . . .	30
edgeDeletedSubgraphs . . . . .	32
efficiency . . . . .	32
eigenvalueBased . . . . .	33
energy . . . . .	35
estrada . . . . .	36
extendedAdjacencyMatrix . . . . .	37
geometricArithmetic1 . . . . .	38
geometricArithmetic2 . . . . .	39
geometricArithmetic3 . . . . .	40
getLabels . . . . .	41
getLargestSubgraph . . . . .	41
globalClusteringCoeff . . . . .	42
graphIndexComplexity . . . . .	43
graphVertexComplexity . . . . .	44
harary . . . . .	45
huXuID . . . . .	46
hyperDistancePathIndex . . . . .	47
infoTheoreticGCM . . . . .	48
konstantinova . . . . .	49
laplaceMatrix . . . . .	50
laplacianEnergy . . . . .	51
laplacianEstrada . . . . .	52
localClusteringCoeff . . . . .	53
meanDistanceDeviation . . . . .	54
mediumArticulation . . . . .	55
minBalabanID . . . . .	56
minConnectivityID . . . . .	57
modifiedZagreb . . . . .	58
narumiKatayama . . . . .	59
normalizedEdgeComplexity . . . . .	60
offdiagonal . . . . .	61
oneEdgeDeletedSubgraphComplexity . . . . .	62
primeID . . . . .	63
productOfRowSums . . . . .	64
radialCentric . . . . .	65

randic . . . . .	66
randomWalkMatrix . . . . .	67
spanningTreeSensitivity . . . . .	68
spectralRadius . . . . .	69
topologicalInfoContent . . . . .	69
totalAdjacency . . . . .	70
twoEdgesDeletedSubgraphComplexity . . . . .	71
variableZagreb . . . . .	72
vertConnectMatrix . . . . .	73
vertexDegree . . . . .	74
weightedID . . . . .	75
weightStrucFuncMatrix_exp . . . . .	76
weightStrucFuncMatrix_lin . . . . .	77
wiener . . . . .	78
zagreb1 . . . . .	79
zagreb2 . . . . .	80
<b>Index</b>	<b>81</b>

---

 QuACN-package

*QuACN: Quantitative Analysis of Complex Networks*


---

## Description

This package offers a set of topological network measures to analyze complex Networks structurally.

## Author(s)

Laurin Mueller

## References

L.A.J. Mueller, K.G. Kugler, A. Dander, A. Graber, M. Dehmer, QuACN: An R Package for Analyzing Complex Biological networks quantitatively, *Bioinformatics*, Vol. 27 no. 1 2011, pages 140-141.

## Examples

```
rm(list=ls())
library("graph")
library("RBGL")
library("QuACN")

###
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)
###
```

```
mat.adj <- adjacencyMatrix(g)
mat.dist <- distanceMatrix(g)
vec.degree <- graph::degree(g)
ska.dia <- diameter(g)
ska.dia <- diameter(g,mat.dist)
##

wien <- wiener(g)
wiener(g,mat.dist)
##
harary(g)
harary(g,mat.dist)
##
balabanJ(g)
balabanJ(g,mat.dist)
##
meanDistanceDeviation(g)
meanDistanceDeviation(g,mat.dist)
##
compactness(g)
compactness(g,mat.dist)
compactness(g,mat.dist,wiener(g,mat.dist))
##
productOfRowSums(g,log=FALSE)
productOfRowSums(g,log=TRUE)
productOfRowSums(g,mat.dist,log=FALSE)
productOfRowSums(g,mat.dist,log=TRUE)
##
hyperDistancePathIndex(g)
hyperDistancePathIndex(g,mat.dist)
hyperDistancePathIndex(g,mat.dist,wiener(g,mat.dist))
##
totalAdjacency(g)
totalAdjacency(g,mat.adj)
##
zagreb1(g)
zagreb1(g,vec.degree)
##
zagreb2(g)
zagreb2(g,vec.degree)
##
randic(g)
randic(g,vec.degree)
##
normalizedEdgeComplexity(g)
normalizedEdgeComplexity(g,totalAdjacency(g,mat.adj))
##
complexityIndexB(g)
complexityIndexB(g,mat.dist)
complexityIndexB(g,mat.dist,vec.degree)
##
degreeDistribution(g)
```

```
degreeDistribution(g,vec.degree)
##
numNodes(g)
numEdges(g)
##
localClusteringCoeff(g)
localClusteringCoeff(g,vec.degree)
##
topologicalInfoContent(g)
topologicalInfoContent(g,mat.dist)
topologicalInfoContent(g,mat.dist,vec.degree)
##
bertz(g)
bertz(g,mat.dist)
bertz(g,mat.dist,vec.degree)
##
bonchev1(g)
bonchev1(g,mat.dist)
##
bonchev2(g)
bonchev2(g,mat.dist)
bonchev2(g,mat.dist,wiener(g))
##
radialCentric(g)
radialCentric(g,mat.dist)
##
vertexDegree(g)
vertexDegree(g,vec.degree)
##
balabanlike1(g)
balabanlike1(g,mat.dist)
##
balabanlike2(g)
balabanlike2(g,mat.dist)
##
graphVertexComplexity(g)
graphVertexComplexity(g,mat.dist)
##
infoTheoreticGCM(g)
infoTheoreticGCM(g,mat.dist,coeff="lin",infofunct="sphere",lambda=1000)
infoTheoreticGCM(g,mat.dist,coeff="exp",infofunct="sphere",lambda=1000)
infoTheoreticGCM(g,mat.dist,coeff="const",infofunct="pathlength",lambda=4000)
infoTheoreticGCM(g,mat.dist,coeff="quad",infofunct="vertcent",lambda=1000)
infoTheoreticGCM(g,mat.dist,coeff="lin",infofunct="degree",lambda=1000)
##
eigenvalueBased(g,adjacencyMatrix,2)
eigenvalueBased(g,laplaceMatrix,2)
eigenvalueBased(g,distanceMatrix,2)
eigenvalueBased(g,distancePathMatrix,2)
eigenvalueBased(g,augmentedMatrix,2)
eigenvalueBased(g,extendedAdjacencyMatrix,2)
eigenvalueBased(g,vertConnectMatrix,2)
eigenvalueBased(g,randomWalkMatrix,2)
```

```
eigenvalueBased(g,weightStrucFuncMatrix_lin,2)  
eigenvalueBased(g,weightStrucFuncMatrix_exp,2)
```

---

adjacencyMatrix      *Adjacency Matrix*

---

### Description

This method calculates the adjacency Matrix of a given graph.

### Usage

```
adjacencyMatrix(g)
```

### Arguments

`g`                    a graph as a graphNEL object.

### Value

This method returns the adjacency Matrix of a given graph.

### Author(s)

Laurin Mueller

### References

F. Harary, Graph Theory, Addison-Wesley series in mathematics, Perseus Books, 1994.

### Examples

```
library(RBGL)  
set.seed(123)  
g <- randomGraph(1:8, 1:5, 0.36)  
  
adjacencyMatrix(g)
```

---

atomBondConnectivity *Atom-bond connectivity (ABC) index*

---

### Description

This method calculates the atom-bond connectivity index.

### Usage

```
atomBondConnectivity(g, deg = NULL)
```

### Arguments

**g** a graph as a graphNEL object.  
**deg** the degree of each node of g. Will be automatically calculated if not supplied.

### Value

This method returns the atom-bond connectivity index of a graph as a double-precision floating point value.

### Author(s)

Lavanya Sivakumar, Michael Schutte

### References

E. Estrada and L. Torres and L. Rodriguez and I. Gutman: An atom-bond connectivity index: Modelling the enthalpy of formation of alkanes. Indian Journal of Chemistry 37A, 1998, 849-855.

### Examples

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:9), 0.5)

# optional: pre-calculate degree of nodes in g
vec.degree <- graph::degree(g)

atomBondConnectivity(g, vec.degree)
```

---

augmentedMatrix	<i>Augmented Matrix</i>
-----------------	-------------------------

---

**Description**

Calculates the augmented vertex degree matrix.

**Usage**

```
augmentedMatrix(g)
```

**Arguments**

g                    A graph as a graphNEL object.

**Details**

for details see the vignette or the reference

**Value**

Aug\_Mat             Returns the augmented vertex degree matrix.

**Author(s)**

Lavanya Sivakumar

**References**

Dehmer M, Sivakumar L, Varmuzua K: Uniquely Discriminating Molecular Structures Using Novel Eigenvalue Based Descriptors. *match* 2012, 67:147-172

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)
augmentedMatrix(g)
```

---

augmentedZagreb	<i>Augmented Zagreb index</i>
-----------------	-------------------------------

---

### Description

This method calculates the augmented Zagreb index.

### Usage

```
augmentedZagreb(g, deg = NULL)
```

### Arguments

<code>g</code>	a graph as a graphNEL object.
<code>deg</code>	the degree of each node of <code>g</code> . Will be automatically calculated if not supplied.

### Value

This method returns the augmented Zagreb index of a graph as a double-precision floating point value.

### Author(s)

Lavanya Sivakumar, Michael Schutte

### References

B. Furtula and A. Graovac and D. Vukicevic: Augmented Zagreb Index. Journal of Mathematical Chemistry, 48:370-380, 2010.

### Examples

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:8), 0.6)

# optional: pre-calculate degree of nodes in g
vec.degree <- graph::degree(g)

augmentedZagreb(g, vec.degree)
```

---

balabanID	<i>Balaban ID number</i>
-----------	--------------------------

---

**Description**

This method calculates the Balaban ID number.

**Usage**

```
balabanID(g, dist=NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the pre-computed distance matrix of the graph. Will be calculated automatically if NULL.

**Value**

The return value is the Balaban ID number of the graph, a weighted path count based on the vertex distance degree.

**Author(s)**

Michael Schutte

**References**

A. T. Balaban. Numerical Modelling of Chemical Structures: Local Graph Invariants and Topological Indices. In Graph Theory and Topology in Chemistry, R. King and D. Rouvray, Eds., pp. 159-176, 1987

**Examples**

```
set.seed(987)
g <- randomEGraph(LETTERS[1:10], 0.3)

balabanID(g)
```

---

`balabanJ`*The Balaban J index*

---

**Description**

This method calculates the Balaban J index.

**Usage**

```
balabanJ(g, dist = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.

**Details**

This method calculates the Balaban J index.

**Value**

It returns the Balaban J index.

**Author(s)**

Laurin Mueller

**References**

A. T. Balaban. Highly discriminating distance-based topological index. Chem.Phys.Lett., 89:383-397, 1991

**Examples**

```
library(graph)
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

#calculate Distance Matrix
mat.dist <- distanceMatrix(g)

balabanJ(g)
```

---

`balabanlike1`*BALABAN-like information index  $U(G)$* 

---

**Description**

This method calculates the BALABAN-like information index  $U(G)$ .

**Usage**

```
balabanlike1(g, dist = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.

**Details**

This method calculates the BALABAN-like information index  $U(G)$ .

**Value**

It return the BALABAN-like information index  $U(G)$ .

**Author(s)**

Laurin Mueller

**References**

A. T. Balaban and T. S. Balaban, New Vertex Invariants and Topological Indices of Chemical Graphs Based on Information on Distances., J. Math. Chem., 1991, 8:383-397

**Examples**

```
library(graph)
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

#calculate Distance Matrix
mat.dist <- distanceMatrix(g)

balabanlike1(g,mat.dist)
```

---

`balabanlike2`*BALABAN-like information index  $XU(G)$* 

---

**Description**

This method calculates the BALABAN-like information index  $X(G)$ .

**Usage**

```
balabanlike2(g, dist = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.

**Details**

This method calculates the BALABAN-like information index  $U(G)$ .

**Value**

It return the BALABAN-like information index  $X(G)$ .

**Author(s)**

Laurin Mueller

**References**

A. T. Balaban and T. S. Balaban, New Vertex Invariants and Topological Indices of Chemical Graphs Based on Information on Distances., J. Math. Chem., 1991, 8:383-397

**Examples**

```
library(graph)
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

#calculate Distance Matrix
mat.dist <- distanceMatrix(g)

balabanlike2(g,mat.dist)
```

---

bertz	<i>Bertz complexity index</i>
-------	-------------------------------

---

**Description**

This method calculates BERTZ complexity index.

**Usage**

```
bertz(g, dist = NULL, deg = NULL)
```

**Arguments**

g	a graph as a graphNEL object.
dist	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.
deg	the degree of each nodes of the graph. If the parameter is empty the degrees will be calculated within the function.

**Details**

This method calculates the BERTZ complexity index.

**Value**

It returns the BERTZ complexity index.

**Author(s)**

Laurin Mueller

**References**

S. H. Bertz. The first general index of molecular complexity. *Journal of the American Chemical Society*, 103:3241-3243, 1981

**Examples**

```
library(graph)
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

#calculate Distance Matrix
mat.dist <- distanceMatrix(g)

bertz(g,mat.dist)
```

---

bonchev1 *Magnitude-based information index by Bonchev I\_D(G)*

---

### Description

This method calculates the magnitude-based information index by Bonchev I\_D(G).

### Usage

```
bonchev1(g, dist = NULL)
```

### Arguments

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.

### Value

It returns the magnitude-based information index by Bonchev I\_D(G)

### Author(s)

Laurin Mueller <laurin@eigenlab.net>

### References

D. Bonchev and N. Trinajstić, Information theory, distance matrix and molecular branching, J. Chem. Phys., 67:4517-4533, 1977

### Examples

```
library(graph)
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

#calculate Distance Matrix
mat.dist <- distanceMatrix(g)

bonchev1(g,mat.dist)
```

---

`bonchev2`*Magnitude-based information index by Bonchev  $I_D^W(G)$* 

---

**Description**

This method calculates the magnitude-based information index by Bonchev  $I_D^W(G)$

**Usage**

```
bonchev2(g, dist = NULL, wien = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.
<code>wien</code>	the Wiener index of <code>g</code> .

**Value**

This method returns the magnitude-based information index by Bonchev  $I_D^W(G)$ .

**Author(s)**

Laurin Mueller <laurin@eigenlab.net>

**References**

D. Bonchev and N. Trinajstić, Information theory, distance matrix and molecular branching, J. Chem. Phys., 67:4517-4533, 1977

**Examples**

```
library(graph)
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

#calculate Distance Matrix
mat.dist <- distanceMatrix(g)

bonchev2(g)
bonchev2(g,mat.dist, wiener(g))
```

---

bondOrderID	<i>Conventional bond order ID number</i>
-------------	--

---

### Description

This method calculates the conventional bond order ID number.

### Usage

```
bondOrderID(g)
```

### Arguments

**g** a graph as a graphNEL object. Each edge must have a "weight" data attribute containing one of the values 1 (single bond), 2 (double bond), 3 (triple bond) or 1.5 (aromatic bond).

### Value

The resulting floating point number is a weighted path sum which takes the different bonds into account.

### Author(s)

Michael Schutte

### References

M. Randić and P. Jurs. On a Fragment Approach to Structure-activity Correlations. *Quantitative Structure-Activity Relationships*, 8(1):39-48, 1989

### Examples

```
set.seed(987)
g <- randomEGraph(LETTERS[1:10], 0.3)

edgeDataDefaults(g, "weight") <- 1
edgeData(g, "B", "I", "weight") <- 2
edgeData(g, "A", "F", "weight") <- 1.5

bondOrderID(g)
```

---

calculateDescriptors    *Generalized descriptor calculation*

---

### Description

The method calculates multiple descriptors for a list of graphs.

### Usage

```
calculateDescriptors(graphs, ..., labels = FALSE)
```

### Arguments

graphs	either a list of or a single graphNEL object.
...	descriptors to calculate and arguments to pass, see ‘Details’.
labels	whether or not the columns of the resulting data frame should be named using the getLabels() method.

### Details

calculateDescriptors() calls each function specified in ‘...’ for every graph in the given list and creates a data frame containing the calculated data. You can specify the functions either as strings (such as “totalAdjacency”) or using the numbers from the following table (e.g., 2001). For convenience, the multiples of 1000 denote entire groups of descriptors.

1000	— all of 1xxx
1001	wiener
1002	harary
1003	balabanJ
1004	meanDistanceDeviation
1005	compactness
1006	productOfRowSums
1007	hyperDistancePathIndex
2000	— all of 2xxx
2001	totalAdjacency
2002	zagreb1
2003	zagreb2
2004	modifiedZagreb
2005	augmentedZagreb
2006	variableZagreb
2007	randic
2008	complexityIndexB
2009	normalizedEdgeComplexity
2010	atomBondConnectivity
2011	geometricArithmetic1
2012	geometricArithmetic2
2013	geometricArithmetic3

2014 narumiKatayama  
3000 — all of 3xxx  
3001 topologicalInfoContent  
3002 bonchev1  
3003 bonchev2  
3004 bertz  
3005 radialCentric  
3006 vertexDegree  
3007 balabanlike1  
3008 balabanlike2  
3009 graphVertexComplexity  
4000 — all of 4xxx  
4001 mediumArticulation  
4002 efficiency  
4003 graphIndexComplexity  
4004 offdiagonal  
4005 spanningTreeSensitivity  
4006 distanceDegreeCentric  
4007 distanceCodeCentric  
5000 — all of 5xxx  
5001 infoTheoreticGCM: vertcent, exp  
5002 infoTheoreticGCM: vertcent, lin  
5003 infoTheoreticGCM: sphere, exp  
5004 infoTheoreticGCM: sphere, lin  
5005 infoTheoreticGCM: pathlength, exp  
5006 infoTheoreticGCM: pathlength, lin  
5007 infoTheoreticGCM: degree, exp  
5008 infoTheoreticGCM: degree, lin  
6000 — all of 6xxx  
6001 eigenvalueBased: adjacencyMatrix, s=1  
6002 eigenvalueBased: adjacencyMatrix, s=2  
6003 eigenvalueBased: laplaceMatrix, s=1  
6004 eigenvalueBased: laplaceMatrix, s=2  
6005 eigenvalueBased: distanceMatrix, s=1  
6006 eigenvalueBased: distanceMatrix, s=2  
6007 eigenvalueBased: distancePathMatrix, s=1  
6008 eigenvalueBased: distancePathMatrix, s=2  
6009 eigenvalueBased: augmentedMatrix, s=1  
6010 eigenvalueBased: augmentedMatrix, s=2  
6011 eigenvalueBased: extendedAdjacencyMatrix, s=1  
6012 eigenvalueBased: extendedAdjacencyMatrix, s=2  
6013 eigenvalueBased: vertConnectMatrix, s=1  
6014 eigenvalueBased: vertConnectMatrix, s=2  
6015 eigenvalueBased: randomWalkMatrix, s=1  
6016 eigenvalueBased: randomWalkMatrix, s=2  
6017 eigenvalueBased: weightStrucFuncMatrix\_lin, s=1  
6018 eigenvalueBased: weightStrucFuncMatrix\_lin, s=2  
6019 eigenvalueBased: weightStrucFuncMatrix\_exp, s=1

```

6020 eigenvalueBased: weightStrucFuncMatrix_exp, s=2
6021 energy
6022 laplacianEnergy
6023 estrada
6024 laplacianEstrada
6025 spectralRadius
7000 — all of 7xxx
7001 oneEdgeDeletedSubgraphComplexity
7002 twoEdgesDeletedSubgraphComplexity
7003 globalClusteringCoeff
8000 — all of 8xxx
8001 connectivityID
8002 minConnectivityID
8003 primeID
8004 bondOrderID
8005 balabanID
8006 minBalabanID
8007 weightedID
8008 huXuID

```

The arguments to these functions, such as the distance matrix or the list of vertex degrees, will be automatically supplied and reused. After each function (or group of functions), regardless of whether it was referred to by name or by its assigned number, you may optionally pass extra arguments as a list, but note that this will not override the calculated arguments. If you wish to pass the same extra arguments to multiple functions, you can concatenate the latter to a vector.

When functions are given by name, an “@NAME” suffix can be used to give the column a different name in the output data frame. This is needed when you want to calculate a descriptor more than once with varying arguments.

### Value

A data frame where rows and columns represent the input graphs and the desired descriptors, respectively. The rows will be named according to the graph list; the column names are the names of the called functions if labels is FALSE, otherwise the label expressions as returned by getLabels() (and found in the vignette).

### Author(s)

Michael Schutte

### Examples

```

library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

calculateDescriptors(g, 1000, 2002, 2003)

```

```

calculateDescriptors(g, "randic", "offdiagonal", 7000, labels=TRUE)

# these will give the same results (although named differently):
calculateDescriptors(g, c(6011, 6013), list(s=3))
calculateDescriptors(g,
  "eigenvalueBased@ea", list(matrix_function="extendedAdjacencyMatrix", s=3),
  "eigenvalueBased@vc", list(matrix_function="vertConnectMatrix", s=3))

```

compactness

*Compactness***Description**

This method calculates the compactness of a graph.

**Usage**

```
compactness(g, dist = NULL, wien = NULL)
```

**Arguments**

<code>g</code>	a graphNEL object
<code>dist</code>	the Distance Matrix of the graph <code>g</code> (optional)
<code>wien</code>	the Wiener index of the graph <code>g</code> (optional)

**Value**

This returns the compactness of the graph.

**Author(s)**

Laurin Mueller

**References**

asdf

**Examples**

```

library(graph)
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

#calculate Distance Matrix
mat.dist <- distanceMatrix(g)

compactness(g)

```

---

complexityIndexB      *The complexity index B*

---

**Description**

This method calculates the complexity index B of a given graph

**Usage**

```
complexityIndexB(g, dist = NULL, deg = NULL)
```

**Arguments**

g	a graph as a graphNEL object.
dist	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.
deg	the degree of each node

**Value**

This returns calculates the complexity index B.

**Author(s)**

Laurin Mueller

**References**

D. Bonchev and D. H. Rouvray, Complexity in Chemistry, Biology, and Ecology, ser. Mathematical and Computational Chemistry. Springer, 2005, New York, NY, USA.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

complexityIndexB(g)
```

---

connectivityID	<i>Randi\`c connectivity ID number</i>
----------------	--

---

### Description

This method calculates the Randi\`c connectivity ID number.

### Usage

```
connectivityID(g, deg=NULL)
```

### Arguments

g	a graph as a graphNEL object.
deg	the degree of each node of g. Will be automatically calculated if not supplied.

### Value

The resulting floating point value is a weighted path sum which stresses local features and takes the vertex degree into account.

### Author(s)

Michael Schutte

### References

M. Randic. On Molecular Identification Numbers. Journal of Chemical Information and Computer Sciences, 24(3):164-175, 1984

### Examples

```
set.seed(987)
g <- randomEGraph(LETTERS[1:10], 0.3)

connectivityID(g)
```

---

degreeDistribution     *Degree Distribution*

---

**Description**

This methods calculates the degree distribution of a given graph.

**Usage**

```
degreeDistribution(g, deg = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>deg</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.

**Value**

This methods returns the degree distribution

**Author(s)**

Laurin Mueller <laurin@eigenlab.net>

**References**

Skorobogatov V.A. and Dobrynin A.A., Metric analysis of graphs, match, pp. 105-151, 1988.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

degreeDistribution(g)
```

---

diameter	<i>Diameter</i>
----------	-----------------

---

**Description**

This method calculates the diameter of a given graph.

**Usage**

```
diameter(g, dist = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.

**Value**

This method returns the diameter of a given graph.

**Author(s)**

Laurin Mueller

**References**

F. Harary, Graph Theory, Addison-Wesley series in mathematics, Perseus Books, 1994.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

diameter(g)
```

---

distanceCodeCentric    *Distance code centric index*

---

### Description

This method calculates the distance code centric index of a graph.

### Usage

```
distanceCodeCentric(g, dist = NULL)
```

### Arguments

**g**                    the input graph as a graphNEL object  
**dist**                 distance matrix of the graph g. Will be automatically calculated if not supplied.

### Value

This returns the distance code centric index of the graph as a double-precision floating point number.

### Author(s)

Lavanya Sivakumar, Michael Schutte

### References

R. Todeschini and V. Consonni and R. Mannhold, Handbook of Molecular Descriptors, Wiley-VCH, Weinheim, Germany, 2002

### Examples

```
library(graph)
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

# calculate Distance Matrix
dist <- distanceMatrix(g)

distanceCodeCentric(g, dist)
```

---

distanceDegreeCentric *Distance degree centric index*

---

### Description

This method calculates the distance degree centric index of a graph.

### Usage

```
distanceDegreeCentric(g, dist = NULL)
```

### Arguments

<code>g</code>	the input graph as a graphNEL object
<code>dist</code>	distance matrix of the graph <code>g</code> . Will be automatically calculated if not supplied.

### Value

This returns the distance degree centric index of the graph as a double-precision floating point number.

### Author(s)

Lavanya Sivakumar, Michael Schutte

### References

R. Todeschini and V. Consonni and R. Mannhold, Handbook of Molecular Descriptors, Wiley-VCH, Weinheim, Germany, 2002

### Examples

```
library(graph)
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

# calculate Distance Matrix
dist <- distanceMatrix(g)

distanceDegreeCentric(g, dist)
```

---

distanceMatrix      *Distance Matrix*

---

**Description**

This method calculates the distance Matrix of a given graph.

**Usage**

```
distanceMatrix(g)
```

**Arguments**

`g`                    a graph as a graphNEL object.

**Value**

This method returns the distance Matrix of a given graph.

**Author(s)**

Laurin Mueller

**References**

F. Harary, Graph Theory, Addison-Wesley series in mathematics, Perseus Books, 1994.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

distanceMatrix(g)
```

---

distancePathMatrix      *Distance Path Matrix*

---

**Description**

Calculates the distance path matrix.

**Usage**

```
distancePathMatrix(g)
```

**Arguments**

g                    A graph as a graphNEL object.

**Details**

for details see the vignette or the reference

**Value**

Dis\_Path\_Mat      Returns distance path matrix.

**Author(s)**

Lavanya Sivakumar

**References**

Dehmer M, Sivakumar L, Varmuzua K: Uniquely Discriminating Molecular Structures Using Novel Eigenvalue Based Descriptors. *match* 2012, 67:147-172

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)
distancePathMatrix(g)
```

---

distSumConnectMatrix    *Distance-sum-connectivity matrix*

---

**Description**

This method calculates the distance-sum-connectivity matrix of a given graph.

**Usage**

```
distSumConnectMatrix(g, dist=NULL)
```

**Arguments**

g                    a graph as a graphNEL object.  
dist                the pre-computed distance matrix of the graph. Will be calculated automatically if NULL.

**Value**

This method returns the distance-sum-connectivity matrix of a given graph. This is an adjacency matrix where each edge is weighted according to the reciprocal square root of the product of the adjacent vertices' distance sum.

**Author(s)**

Michael Schutte

**References**

K. Szymanski, W. Mueller, J. Knop, and N. Trinajstić. On the Identification Numbers for Chemical Structures. *International Journal of Quantum Chemistry*, 30(S20):173-183, 1986

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

distSumConnectMatrix(g)
```

---

dobrynin

*Dobrynin indices*

---

**Description**

This function calculates a set of basic descriptors introduced by Skorobogatov and Dobrynin.

**Usage**

```
dobrynin(g, dist = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.

**Details**

This methods calculates 14 basic descriptors introduced by Skorobogatov and Dobrynin.

**Value**

This function return a list containing the 14 descriptors.

<code>eccentricityVertex</code>	Eccentricity of all vertices
<code>eccentricityGraph</code>	Eccentricity of a graph
<code>avgeccOfG</code>	Average eccentricity of a graph
<code>eccentricVertex</code>	Eccentric of all vertices
<code>eccentricGraph</code>	Eccentric of a graph
<code>vertexCentrality</code>	Vertex centrality
<code>graphIntegration</code>	Graph integration
<code>unipolarity</code>	Unipolarity of a graph
<code>vertexDeviation</code>	Deviation of all vertices
<code>variation</code>	Variation of a graph
<code>centralization</code>	Centralization of a graph
<code>avgDistance</code>	Average distance of a graph
<code>distVertexDeviation</code>	Distance vertex deviation
<code>meanDistVertexDeviation</code>	Mean distance vertex deviation

**Author(s)**

Laurin Mueller

**References**

Skorobogatov V.A. and Dobrynin A.A., Metric analysis of graphs, *match*, pp. 105-151, 1988.

**Examples**

```
library(graph)
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

dobrynin(g)
```

---

edgeDeletedSubgraphs *Edge-deleted subgraphs*

---

**Description**

This method lists all edge-deleted subgraphs of a given graph.

**Usage**

```
edgeDeletedSubgraphs(gs)
```

**Arguments**

gs a list of or a single graph, either as a graphNEL object or as an adjacency matrix.

**Value**

This method returns a flat list of all unique subgraphs which can be constructed from the input graph(s) by removing a single edge. The individual graphs are output as adjacency matrices.

**Author(s)**

Lavanya Sivakumar, Michael Schutte <michi@uiaae.at>

**Examples**

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:8), 0.55)

one.eds <- edgeDeletedSubgraphs(g)
two.eds <- edgeDeletedSubgraphs(one.eds)
```

---

efficiency *Efficiency complexity index*

---

**Description**

This method calculates the efficiency complexity measure.

**Usage**

```
efficiency(g, dist=NULL)
```

**Arguments**

`g` a graph as a graphNEL object.  
`dist` the distance matrix of the graph. Will be calculated automatically if left empty.

**Details**

This method calculates the efficiency complexity measure.

**Value**

It returns the efficiency complexity measure as a double-precision floating point value.

**Author(s)**

Lavanya Sivakumar, Michael Schutte

**References**

J. Kim and T. Wilhelm. What is a complex graph? *Physica A*, 387:2637-2652, 2008

**Examples**

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:10), 0.6)

efficiency(g)

# alternatively:
dist <- distanceMatrix(g)
efficiency(g, dist)
```

---

eigenvalueBased

*Eigenvalue-based Descriptors*

---

**Description**

Eigenvalue-based Descriptors

**Usage**

```
eigenvalueBased(g, matrix_function, s=1)
```

### Arguments

<code>g</code>	A graph as a graphNEL object.
<code>matrix_function</code>	The matrix function to calculate the desired matrix for the graph. For details see the vignette or the example section below.
<code>s</code>	Parameter to calculate the descriptors, see reference. Default set to 1.

### Details

For details see the Vignette.

### Value

It returns a list with following items:

HMs	Formula (2) in the reference paper.
SMstance	Formula (3) in the reference paper.
ISMs	Formula (4) in the reference paper.
PMs	Formula (5) in the reference paper.
IPMs	Formula (6) in the reference paper.

### Author(s)

Lavanya Sivakumar, Laurin Mueller

### References

Dehmer M, Sivakumar L, Varmuzua K: Uniquely Discriminating Molecular Structures Using Novel Eigenvalue Based Descriptors. *match* 2012, 67:147-172

### Examples

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)
mat.dist <- distanceMatrix(g)

##Matrices like in the paper
##1. adjacency matrix
eigenvalueBased(g,adjacencyMatrix,2)
##2. Laplacian matrix
eigenvalueBased(g,laplaceMatrix,2)
##3. Distance matrix
eigenvalueBased(g,distanceMatrix,2)
##4. Distance path Matrix
eigenvalueBased(g,distancePathMatrix,2)
##5. Augmented vertex degree matrix
eigenvalueBased(g,augmentedMatrix,2)
##6. Extended adjacency matrix
```

```
eigenvalueBased(g,extendedAdjacencyMatrix,2)
##7. Connectivity matrix
eigenvalueBased(g,vertConnectMatrix,2)
##8. Random Walk markov matrix
eigenvalueBased(g,randomWalkMatrix,2)
##9. Weighted structure function matrix IM1
eigenvalueBased(g,weightStrucFuncMatrix_lin,2)
##10. Weighted structure function matrix IM2
eigenvalueBased(g,weightStrucFuncMatrix_exp,2)
```

---

energy

*Graph energy*

---

### Description

This method calculates the energy of a graph.

### Usage

```
energy(g)
```

### Arguments

`g` a graph as a graphNEL object.

### Value

This method returns the energy of a graph as a double-precision floating point value.

### Author(s)

Lavanya Sivakumar, Michael Schutte

### Examples

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:6), 0.35)

energy(g)
```

---

estrada	<i>Estrada index</i>
---------	----------------------

---

**Description**

This method calculates the Estrada index of a graph.

**Usage**

```
estrada(g)
```

**Arguments**

`g` a graph as a graphNEL object.

**Value**

This method returns the estrada index of a graph as a double-precision floating point value.

**Author(s)**

Lavanya Sivakumar, Michael Schutte

**References**

E. Estrada: Characterization of 3D molecular structure. *Chemical Physics Letters*, 319:713-718, 2000

**Examples**

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:8), 0.6)

estrada(g)
```

---

`extendedAdjacencyMatrix`*Extended Adjacency Matrix*

---

**Description**

Calculates the extended adjacency matrix.

**Usage**

```
extendedAdjacencyMatrix(g)
```

**Arguments**

`g` A graph as a graphNEL object.

**Details**

for details see the vignette or the reference

**Value**

`ExtAdjMat` Returns the extended adjacency matrix.

**Author(s)**

Lavanya Sivakumar

**References**

Dehmer M, Sivakumar L, Varmuzua K: Uniquely Discriminating Molecular Structures Using Novel Eigenvalue Based Descriptors. *match* 2012, 67:147-172

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)
extendedAdjacencyMatrix(g)
```

geometricArithmetic1 *First geometric-arithmetic index*

---

### Description

This method calculates the first geometric-arithmetic (GA1) index.

### Usage

```
geometricArithmetic1(g, deg = NULL)
```

### Arguments

`g` a graph as a graphNEL object.  
`deg` the degree of each node of `g`. Will be automatically calculated if not supplied.

### Value

This method returns the first geometric-arithmetic index of a graph as a double-precision floating point value.

### Author(s)

Lavanya Sivakumar, Michael Schutte

### References

B. Zhou and I. Gutman and B. Furtula and Z. Du: On two types of geometric-arithmetic index. Chemical Physics Letters, 482:153-155, 2009

### Examples

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:9), 0.5)

# optional: pre-calculate degree of nodes in g
vec.degree <- graph::degree(g)

geometricArithmetic1(g, vec.degree)
```

---

geometricArithmetic2 *Second geometric-arithmetic index*

---

### Description

This method calculates the second geometric-arithmetic (GA2) index.

### Usage

```
geometricArithmetic2(g, dist = NULL)
```

### Arguments

**g** a graph as a graphNEL object.  
**dist** the distance matrix of g. Will be automatically calculated if not supplied.

### Value

This method returns the second geometric-arithmetic index of a graph as a double-precision floating point value.

### Author(s)

Lavanya Sivakumar, Michael Schutte

### References

B. Zhou and I. Gutman and B. Furtula and Z. Du: On two types of geometric-arithmetic index. Chemical Physics Letters, 482:153-155, 2009

### Examples

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:9), 0.5)

# optional: pre-calculate distance matrix
mat.dist <- distanceMatrix(g)

geometricArithmetic2(g, mat.dist)
```

---

geometricArithmetic3 *Third geometric-arithmetic index*

---

### Description

This method calculates the third geometric-arithmetic (GA3) index.

### Usage

```
geometricArithmetic3(g, dist = NULL)
```

### Arguments

**g** a graph as a graphNEL object.  
**dist** the distance matrix of g. Will be automatically calculated if not supplied.

### Value

This method returns the third geometric-arithmetic index of a graph as a double-precision floating point value.

### Author(s)

Lavanya Sivakumar, Michael Schutte

### References

B. Zhou and I. Gutman and B. Furtula and Z. Du: On two types of geometric-arithmetic index. Chemical Physics Letters, 482:153-155, 2009

### Examples

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:9), 0.5)

# optional: pre-calculate distance matrix
mat.dist <- distanceMatrix(g)

geometricArithmetic3(g, mat.dist)
```

---

`getLabels`*This method returns the label of a descriptor as expression.*

---

**Description**

This method returns the label of a descriptor as expression.

**Usage**

```
getLabels(1)
```

**Arguments**

1

**Value**

This method returns the label of a descriptor as expression.

**Author(s)**

Laurin Mueller

**Examples**

```
getLabels("wiener")
```

---

`getLargestSubgraph`*A function to extract the largest subgraph from a graphNEL object*

---

**Description**

In QuACN most methods depend on the analyzed graph to be connected. This function extracts the largest connected component from a graphNEL object.

**Usage**

```
getLargestSubgraph(g)
```

**Arguments**

g

A graphNEL object of which the largest connected component has to be extracted.

**Value**

The largest connected graphNEL object from g

**Note**

Code taken from Hahne et al. "Bioconductor Case Studies"

**Author(s)**

Karl Kugler

**References**

Florian Hahne, Wolfgang Huber, Robert Gentleman, Seth Falcon "Bioconductor Case Studies", Springer, 2008

**Examples**

```
set.seed(667)
g <- randomGraph(paste("A",1:100, sep=""), 1:4, p=0.03)
lcc <- getLargestSubgraph(g)
lcc
```

---

globalClusteringCoeff *Global Clustering Coefficient*

---

**Description**

This method calculates the Global Clustering Coefficient.

**Usage**

```
globalClusteringCoeff(g, loc = NULL)
```

**Arguments**

g	a graph as a graphNEL object.
loc	the the local clustering coefficient.

**Value**

This method returns the global clustering coefficient.

**Author(s)**

Laurin Mueller

**References**

D. Watts, Small Worlds: The Dynamics of Networks Between Order and Randomness. Princeton Univ Pr, 2003. D. Watts and S. Strogatz, Collective dynamics of 'Small-World' Networks, Nature, vol. 393, no. 6684, pp. 440-442, 1998.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

loccc <- localClusteringCoeff(g)
globalClusteringCoeff(g)
globalClusteringCoeff(g,loc=loccc)
```

---

graphIndexComplexity *Graph index complexity measure*

---

**Description**

This method calculates the graph index complexity measure.

**Usage**

```
graphIndexComplexity(g)
```

**Arguments**

**g** a graph as a graphNEL object.

**Details**

This method calculates the graph index complexity measure.

**Value**

It returns the graph index complexity measure as a double-precision floating point value.

**Author(s)**

Lavanya Sivakumar, Michael Schutte

**References**

J. Kim and T. Wilhelm. What is a complex graph? Physica A, 387:2637-2652, 2008

**Examples**

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:10), 0.6)

graphIndexComplexity(g)
```

---

graphVertexComplexity *Graph Vertex Complexity*

---

**Description**

This method calculates the Graph Vertex Complexity.

**Usage**

```
graphVertexComplexity(g, dist = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.

**Value**

This method returns the Graph Vertex Complexity.

**Author(s)**

Laurin Mueller

**References**

C. Raychaudhury, S. K. Ray, J. J. Ghosh, A. B. Roy, and S. C. Basak, Discrimination of Isomeric Structures using Information Theoretic Topological Indices, *Journal of Computational Chemistry*, vol. 5, pp. 581-588, 1984.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

graphVertexComplexity(g)
```

---

harary

*Harary Index*

---

### Description

This method calculates the Harary Index.

### Usage

```
harary(g, dist = NULL)
```

### Arguments

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.

### Details

This method calculates the Harary index:

### Value

It returns the Harary Index.

### Author(s)

Laurin Mueller

### References

A. T. Balaban and O. Ivanciuc, Historical Development of Topological Indices, in Topological Indices and Related Descriptors in QSAR and QSPAR, J. Devillers and A. T. Balaban, Eds. Gordon and Breach Science Publishers, 1999, pp. 21-57, Amsterdam, The Netherlands.

### Examples

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

harary(g)
```

---

huXuID

*Hu-Xu ID number*

---

### Description

This method calculates the Hu-Xu ID number.

### Usage

```
huXuID(g, deg=NULL)
```

### Arguments

**g** a graph as a graphNEL object. Each edge must have a "weight" data attribute containing one of the values 1 (single bond), 2 (double bond), 3 (triple bond) or 1.5 (aromatic bond), and each vertex must have an "atom" data attribute specifying its atomic number.

**deg** the degree of each node of g. Will be automatically calculated if not supplied.

### Value

The resulting floating point value is computed from weighted path sums based on the vertex degree, the atomic numbers, and edge multiplicity.

### Author(s)

Michael Schutte

### References

C. Hu and L. Xu. On Hall and Kier's Topological State and Total Topological Index. *Journal of Chemical Information and Computer Sciences*, 34(6):1251-1258, 1994

### Examples

```
set.seed(987)
g <- randomEGraph(LETTERS[1:10], 0.3)

edgeDataDefaults(g, "weight") <- 1
edgeData(g, "B", "I", "weight") <- 2
edgeData(g, "A", "F", "weight") <- 1.5

nodeDataDefaults(g, "atom") <- 6
nodeData(g, "A", "atom") <- 8

huXuID(g)
```

---

`hyperDistancePathIndex`*Hyper Distance Path Index*

---

**Description**

This method calculates the Hyper Distance Path Index.

**Usage**

```
hyperDistancePathIndex(g, dist = NULL, wien = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.
<code>wien</code>	the wiener index of <code>g</code> .

**Value**

This method returns the Hyper Distance Path Index.

**Author(s)**

Laurin Mueller <laurin@eigenlab.net>

**References**

R. Todeschini, V. Consonni, and R. Mannhold, Handbook of Molecular Descriptors. Weinheim, Germany. Wiley-VCH, 2002.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

degreeDistribution(g)
```

---

infoTheoreticGCM      *Information theoretic graph complexity measures*

---

### Description

Measures of this group assign a probability value to each vertex of the network using a so-called information functional  $f$  which captures structural information of the network  $g$ . Note that some combinations of the settings can cause the descriptor to return NaN. In that case you have to check for warnings.

### Usage

```
infoTheoreticGCM(g, dist = NULL, coeff = "lin", infofunct = "sphere",
lambda = 1000, custCoeff=NULL, alpha=0.5, prec=53)
```

### Arguments

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.
<code>coeff</code>	specifies the weighting coefficient. Possible values are "lin" (default), "quad", "exp", "const" or "cust". If it is set to "cust" you have to specify your customized weighting schema with the parameter <code>custCoeff</code> .
<code>infofunct</code>	specifies the information functional. Possible values are "sphere" (default), "pathlength", "vertcent" or "degree" .
<code>lambda</code>	specifies the scaling constant for the distance measures. The default value is 1000.
<code>custCoeff</code>	specifies the customized weighting schema. To use it you need to set <code>coeff="const"</code> .
<code>alpha</code>	alpha for degree degree association.
<code>prec</code>	specifies the floating-point precision to use (currently only implemented for degree-degree association). Values up to 53 are handled with the built-in double data type; larger values trigger the usage of Rmpfr.

### Details

For details see the vignette.

### Value

The returned list consists of the following items:

<code>entropy</code>	contains the calculated entropy measure.
<code>distance</code>	contains the calculated distance measure.
<code>pis</code>	contains the calculated probability distribution.
<code>fvi</code>	contains the calculated values of the information functional, for each vertex.

If any of these values is NaN, please check if your parameters are valid. For `infofunct="degree"` in particular, the result might be impossible to represent using a standard R numeric vector. In this case the `"prec"` parameter has to be set to a higher value.

If `infofunct` is `"degree"` and `prec` is greater than 53, the resulting values will be of class `"mpfr"` (instead of `"numeric"` in all other cases). Note that if you use such a vector in a calculation, arbitrary precision floating point arithmetics will be used throughout, even if the other operands are regular double values. You can use `"as.double"` at any point to convert an `"mpfr"` vector to the built-in `"numeric"` class (losing precision).

### Author(s)

Laurin Mueller

### References

M. Dehmer, Information processing in complex networks: Graph entropy and information functionals, *Applied Mathematics and Computation*, 202:82-94, 2008

Dehmer M., Emmert-Streib F., Tsoy R. Y., Varmuza K.: Quantifying Structural Complexity of Graphs: Information Measures in Mathematical Chemistry. In: Putz M. (Editor): *Quantum Frontiers of Atoms and Molecules in Physics, Chemistry, and Biology*, Nova Science Publishers, to appear, 2010

### Examples

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)
mat.dist <- distanceMatrix(g)

infoTheoreticGCM(g)
infoTheoreticGCM(g,mat.dist,coeff="lin",infofunct="sphere",lambda=1000)
infoTheoreticGCM(g,mat.dist,coeff="const",infofunct="pathlength",lambda=4000)
infoTheoreticGCM(g,mat.dist,coeff="quad",infofunct="vertcent",lambda=1000)
infoTheoreticGCM(g,mat.dist,coeff="exp",infofunct="degree",lambda=1000)
```

---

konstantinova

*Konstantinova*

---

### Description

This method calculates the Konstantinova index

### Usage

```
konstantinova(g, dist = NULL)
```

**Arguments**

`g` a graphNEL object  
`dist` the Distance Matrix of the graph `g` (optional)

**Value**

It returns the Konstantinova index.

**Author(s)**

Andreas Dander <andreas.dander@umit.at>  
Laurin Mueller

**References**

E. V. Konstantinova and A. A. Paleev. Sensitivity of topological indices of polycyclic graphs. Vychisl. Sistemy, 136:38-48, 1990, In Russian.

**Examples**

```
library(QuACN)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)
mat.dist <- distanceMatrix(g)

konstantinova(g)
konstantinova(g,dist=mat.dist)
```

---

laplaceMatrix

*Laplacian Matrix*

---

**Description**

Calculates the laplacian matrix.

**Usage**

```
laplaceMatrix(g)
```

**Arguments**

`g` A graph as a graphNEL object.

**Details**

for deatils see the vignette or the reference

**Value**

Lap\_Mat            Returns the laplacian matrix.

**Author(s)**

Lavanya Sivakumar

**References**

Dehmer M, Sivakumar L, Varmuzua K: Uniquely Discriminating Molecular Structures Using Novel Eigenvalue Based Descriptors. *match* 2012, 67:147-172

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)
laplaceMatrix(g)
```

---

laplacianEnergy            *Laplacian energy of a graph*

---

**Description**

This method calculates the Laplacian energy of a graph.

**Usage**

```
laplacianEnergy(g)
```

**Arguments**

g                    a graph as a graphNEL object.

**Value**

This method returns the Laplacian energy of a graph as a double-precision floating point value.

**Author(s)**

Lavanya Sivakumar, Michael Schutte

**References**

I. Gutman and B. Zhou: Laplacian energy of a graph. *Linear Algebra and its Applications*, 414:29-37, 2006.

**Examples**

```
library(graph)
set.seed(123)
g <- randomERGraph(as.character(1:8), 0.6)

laplacianEnergy(g)
```

---

laplacianEstrada      *Laplacian Estrada index*

---

**Description**

This method calculates the Laplacian Estrada index of a graph.

**Usage**

```
laplacianEstrada(g)
```

**Arguments**

`g`                    a graph as a graphNEL object.

**Value**

This method returns the Laplacian Estrada index of a graph as a double-precision floating point value.

**Author(s)**

Lavanya Sivakumar, Michael Schutte

**References**

G. H. Fath-Tabar and A. R. Ashrafi and I. Gutman: Note on Estrada and L-Estrada indices of graphs, Bull. Cl. Sci. Math. Nat. Sci. Math. CXXXIX, 2009

**Examples**

```
library(graph)
set.seed(123)
g <- randomERGraph(as.character(1:8), 0.6)

laplacianEstrada(g)
```

---

localClusteringCoeff *Local Clustering Coefficient*

---

**Description**

This method calculates the Local Clustering Coefficient.

**Usage**

```
localClusteringCoeff(g, deg = NULL)
```

**Arguments**

**g** a graph as a graphNEL object.  
**deg** the degree of each node of the g.

**Value**

This method returns the local clustering coefficient.

**Author(s)**

Laurin Mueller

**References**

D. Watts, Small Worlds: The Dynamics of Networks Between Order and Randomness. Princeton Univ Pr, 2003. D. Watts and S. Strogatz, Collective dynamics of 'Small-World' Networks, Nature, vol. 393, no. 6684, pp. 440-442, 1998.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

deg <- graph::degree(g)
localClusteringCoeff(g)
localClusteringCoeff(g, deg)
```

---

meanDistanceDeviation *Mean Distance Deviation*

---

**Description**

This method calculates the Mean Distance Deviation.

**Usage**

```
meanDistanceDeviation(g, dist = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.

**Value**

This method returns Mean Distance Deviation.

**Author(s)**

Laurin Mueller <laurin@eigenlab.net>

**References**

Skorobogatov V.A. and Dobrynin A.A., Metric analysis of graphs, *match*, pp. 105-151, 1988.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

meanDistanceDeviation(g)
```

---

mediumArticulation	<i>Medium articulation index</i>
--------------------	----------------------------------

---

**Description**

This method calculates the medium articulation index.

**Usage**

```
mediumArticulation(g)
```

**Arguments**

`g` a graph as a graphNEL object.

**Details**

This method calculates the medium articulation index.

**Value**

It returns the medium articulation index as a double-precision floating point value.

**Author(s)**

Lavanya Sivakumar, Michael Schutte

**References**

J. Kim and T. Wilhelm. What is a complex graph? *Physica A*, 387:2637-2652, 2008

**Examples**

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:10), 0.6)

mediumArticulation(g)
```

---

minBalabanID	<i>Balaban ID number considering shortest paths only</i>
--------------	--

---

### Description

This method calculates a modified, faster version of the Balaban ID number.

### Usage

```
minBalabanID(g, dist=NULL)
```

### Arguments

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the pre-computed distance matrix of the graph. Will be calculated automatically if NULL.

### Value

This method works like the `balabanID` method, but it only sums the weights of all the shortest paths in the graph. This results in different values only if the graph contains cycles.

### Author(s)

Michael Schutte

### References

O. Ivanciuc and A. Balaban. Design of Topological Indices. Part 3. New Identification Numbers for Chemical Structures: MINID and MINSID. *Croatica chemica acta*, 69:9-16, 1996

### Examples

```
set.seed(987)
g <- randomEGraph(LETTERS[1:10], 0.3)

minBalabanID(g)
```

---

minConnectivityID	<i>Connectivity ID number considering shortest paths only</i>
-------------------	---

---

**Description**

This method calculates a modified, faster version of the connectivity ID number.

**Usage**

```
minConnectivityID(g, deg=NULL)
```

**Arguments**

**g** a graph as a graphNEL object.  
**deg** the degree of each node of g. Will be automatically calculated if not supplied.

**Value**

This method works like the connectivityID method, but it only sums the weights of all the shortest paths in the graph. This results in different values only if the graph contains cycles.

**Author(s)**

Michael Schutte

**References**

O. Ivanciuc and A. Balaban. Design of Topological Indices. Part 3. New Identification Numbers for Chemical Structures: MINID and MINSID. *Croatica chemica acta*, 69:9-16, 1996

**Examples**

```
set.seed(987)
g <- randomEGraph(LETTERS[1:10], 0.3)

minConnectivityID(g)
```

---

modifiedZagreb	<i>Modified Zagreb index</i>
----------------	------------------------------

---

**Description**

This method calculates the modified Zagreb index.

**Usage**

```
modifiedZagreb(g, deg = NULL)
```

**Arguments**

`g` a graph as a graphNEL object.  
`deg` the degree of each node of `g`. Will be automatically calculated if not supplied.

**Value**

This method returns the modified Zagreb index of a graph as a double-precision floating point value.

**Author(s)**

Lavanya Sivakumar, Michael Schutte

**References**

S. Nikolic and G. Kovacevic and A. Milicevic and N. Trinajstic: The Zagreb Indices 30 Years After. *Croatica Chemica Acta*, 76:113-124, 2003

**Examples**

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:9), 0.5)

# optional: pre-calculate degree of nodes in g
vec.degree <- graph::degree(g)

modifiedZagreb(g, vec.degree)
```

---

narumiKatayama	<i>Narumi-Katayama index</i>
----------------	------------------------------

---

**Description**

This method calculates the Narumi-Katayama index.

**Usage**

```
narumiKatayama(g, deg = NULL)
```

**Arguments**

g	a graph as a graphNEL object.
deg	the degree of each node of g. Will be automatically calculated if not supplied.

**Value**

This method returns the Narumi-Katayama index of a graph as an integer value.

**Author(s)**

Lavanya Sivakumar, Michael Schutte

**References**

H. Narumi and M. Katayama: Simple topological index. A newly devised index characterizing the topological nature of structural isomers of saturated hydrocarbons. Mem. Fac. Engin. Hokkaido Univ., 16:209, 1984

**Examples**

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:9), 0.5)

# optional: pre-calculate degree of nodes in g
vec.degree <- graph::degree(g)

narumiKatayama(g, vec.degree)
```

---

normalizedEdgeComplexity  
*Normalized Edge Complexity*

---

**Description**

This method calculates the Normalized Edge Complexity.

**Usage**

```
normalizedEdgeComplexity(g, ita = NULL)
```

**Arguments**

`g` a graph as a graphNEL object.  
`ita` the total adjacency measure.

**Value**

This method returns the Normalized Edge Complexity

**Author(s)**

Laurin Mueller <laurin@eigenlab.net>

**References**

D. Bonchev and D. H. Rouvray, Complexity in Chemistry, Biology, and Ecology, ser. Mathematical and Computational Chemistry. Springer, 2005, New York, NY, USA.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

normalizedEdgeComplexity(g)
```

---

`offdiagonal`*Offdiagonal complexity index*

---

**Description**

This method calculates the offdiagonal complexity measure.

**Usage**

```
offdiagonal(g, deg = NULL)
```

**Arguments**

`g` a graph as a graphNEL object.  
`deg` the degree of all nodes of `g`. Will be calculated automatically if left empty.

**Details**

This method calculates the offdiagonal complexity measure.

**Value**

It returns the offdiagonal complexity measure as a double-precision floating point value.

**Author(s)**

Lavanya Sivakumar, Michael Schutte

**References**

J. Kim and T. Wilhelm. What is a complex graph? *Physica A*, 387:2637-2652, 2008

**Examples**

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:10), 0.6)

offdiagonal(g)

# alternatively:
deg <- graph::degree(g)
offdiagonal(g, deg)
```

---

`oneEdgeDeletedSubgraphComplexity`*One-edge-deleted subgraph complexity measures*

---

**Description**

This method calculates two indices based on one-edge-deleted subgraphs.

**Usage**

```
oneEdgeDeletedSubgraphComplexity(g, one.eds = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>one.eds</code>	the one-edge-deleted subgraphs of <code>g</code> as a list of adjacency matrices, as returned by <code>edgeDeletedSubgraphs(g)</code> . If this parameter is omitted, the subgraphs will be calculated automatically.

**Details**

This method calculates the one-edge-deleted subgraph complexity with respect to the different number of spanning trees (`C_1eST`) and spectra of the Laplacian and signless Laplacian matrix (`C_1eSpec`).

**Value**

The results are returned in a list with two entries named `C_1eST` and `C_1eSpec`.

**Author(s)**

Lavanya Sivakumar, Michael Schutte

**References**

J. Kim and T. Wilhelm. What is a complex graph? *Physica A*, 387:2637-2652, 2008

**Examples**

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:12), 0.5)

result <- oneEdgeDeletedSubgraphComplexity(g)
result$C_1eST
result$C_1eSpec
```

---

primeID	<i>Prime ID number</i>
---------	------------------------

---

**Description**

This method calculates the prime ID number.

**Usage**

```
primeID(g, deg=NULL)
```

**Arguments**

**g** a graph as a graphNEL object.  
**deg** the degree of each node of g. Will be automatically calculated if not supplied.

**Value**

This method works like the connectivityID method, but it assigns distinct prime numbers to each different pair of vertex degrees associated with an edge.

**Author(s)**

Michael Schutte

**References**

M. Randic. Molecular ID numbers: By Design. Journal of Chemical Information and Computer Sciences, 26(3):134-136, 1986

**Examples**

```
set.seed(987)
g <- randomEGraph(LETTERS[1:10], 0.3)

primeID(g)
```

---

productOfRowSums      *Product of Row Sums*

---

### Description

This method calculates the product of row sums.

### Usage

```
productOfRowSums(g, dist = NULL, log = FALSE)
```

### Arguments

**g**                    a graph as a graphNEL object.

**dist**                the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.

**log**                if TRUE it returns the log of the product. The default value is FALSE.

### Value

This method returns the product of row sums.

### Author(s)

Laurin Mueller

### References

H. P. Schultz, E. B. Schultz, and T. P. Schultz, Topological Organic Chemistry. 4. Graph Theory, Matrix Permanents, and Topological Indices of Alkanes, Journal of Chemical Information and Computer Sciences, vol. 32, no. 1, pp. 69-72, 1992.

### Examples

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

productOfRowSums(g)
```

---

radialCentric	<i>Radial Centric Information Index</i>
---------------	---

---

**Description**

This method calculates the Radial Centric Information Index.

**Usage**

```
radialCentric(g, dist = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.

**Value**

This method returns the Radial Centric Information Index.

**Author(s)**

Laurin Mueller <laurin@eigenlab.net>

**References**

D. Bonchev, Information Theoretic Indices for Characterization of Chemical Structures. Research Studies Press, Chichester, 1983.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

radialCentric(g)
```

---

randic	<i>Randic connectivity index</i>
--------	----------------------------------

---

**Description**

This method calculates the Randic connectivity index.

**Usage**

```
randic(g, deg = NULL)
```

**Arguments**

g	a graph as a graphNEL object.
deg	the degree of each node of g.

**Value**

This method returns the Randic connectivity index.

**Author(s)**

Laurin Mueller <laurin@eigenlab.net>

**References**

X. Li and I. Gutman, Mathematical Aspects of Randić-Type Molecular Structure Descriptors, ser. Mathematical Chemistry Monographs. University of Kragujevac and Faculty of Science Kragujevac, 2006.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

randic(g)
```

---

randomWalkMatrix	<i>Random Walk Markov Matrix</i>
------------------	----------------------------------

---

### Description

Calculates the random walk markov matrix.

### Usage

```
randomWalkMatrix(g)
```

### Arguments

`g` A graph as a graphNEL object.

### Details

for deatils see the vignette or the reference

### Value

`RanWalk_Mat` Returns the random walk markov matrix.

### Author(s)

Lavanya Sivakumar

### References

Dehmer M, Sivakumar L, Varmuzua K: Uniquely Discriminating Molecular Structures Using Novel Eigenvalue Based Descriptors. *match* 2012, 67:147-172

### Examples

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)
randomWalkMatrix(g)
```

---

`spanningTreeSensitivity`*Spanning tree sensitivity measures*

---

**Description**

This method calculates two spanning tree sensitivity measures.

**Usage**

```
spanningTreeSensitivity(g, one.eds = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>one.eds</code>	the one-edge-deleted subgraphs of <code>g</code> as a list of adjacency matrices, as returned by <code>edgeDeletedSubgraphs(g)</code> . If this parameter is omitted, the subgraphs will be calculated automatically.

**Details**

This method calculates the spanning tree sensitivity (STS) and the spanning tree sensitivity differences (STSD) measures.

**Value**

The results are returned in a list with two entries named STS and STSD.

**Author(s)**

Lavanya Sivakumar, Michael Schutte

**References**

J. Kim and T. Wilhelm. What is a complex graph? *Physica A*, 387:2637-2652, 2008

**Examples**

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:12), 0.5)

result <- spanningTreeSensitivity(g)
result$STS
result$STSD
```

---

spectralRadius	<i>Spectral radius</i>
----------------	------------------------

---

**Description**

This method calculates the spectral radius of a graph.

**Usage**

```
spectralRadius(g)
```

**Arguments**

`g` a graph as a graphNEL object.

**Value**

This method returns the spectral radius of a graph as a double-precision floating point value.

**Author(s)**

Lavanya Sivakumar, Michael Schutte

**Examples**

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:9), 0.5)

spectralRadius(g)
```

---

topologicalInfoContent	<i>Topological Information Conten</i>
------------------------	---------------------------------------

---

**Description**

This method calculates the Topological Information Content.

**Usage**

```
topologicalInfoContent(g, dist = NULL, deg = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.
<code>deg</code>	the degree of each node of <code>g</code> .

**Value**

This method returns the Topological Information Content.

**Author(s)**

Laurin Mueller <laurin@eigenlab.net>

**References**

A. Mowshowitz, Entropy and the Complexity of the Graphs I: An Index of the Relative Complexity of a Graph, *Bull. Math. Biophys.*, vol. 30, pp. 175-204, 1968. N. Rashevsky, Life, Information Theory, and Topology, *Bull. Math. Biophys.*, vol. 17, pp. 229-235, 1955.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

topologicalInfoContent(g)
```

---

<code>totalAdjacency</code>	<i>Index of total Adjacency</i>
-----------------------------	---------------------------------

---

**Description**

This method calculates the Index of total Adjacency.

**Usage**

```
totalAdjacency(g, am = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>am</code>	the adjacency matrix of <code>g</code> .

**Value**

This method returns

**Author(s)**

Laurin Mueller

**References**

D. Bonchev and D. H. Rouvray, Complexity in Chemistry, Biology, and Ecology, ser. Mathematical and Computational Chemistry. Springer, 2005, New York, NY, USA.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

totalAdjacency(g)
```

---

twoEdgesDeletedSubgraphComplexity

*Two-edges-deleted subgraph complexity*

---

**Description**

This method calculates the two-edges-deleted subgraph complexity based on Laplacian matrices.

**Usage**

```
twoEdgesDeletedSubgraphComplexity(g, two.eds = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>two.eds</code>	the two-edges-deleted subgraphs of <code>g</code> as a list of adjacency matrices, as returned by <code>edgeDeletedSubgraphs(edgeDeletedSubgraphs(g))</code> . If this parameter is omitted, the subgraphs will be calculated automatically.

**Details**

This method calculates the two-edges-deleted subgraph complexity with respect to different spectra of the Laplacian and signless Laplacian matrix.

**Value**

The return value is the described two-edges-deleted subgraph complexity measure as a double-precision floating point number.

**Author(s)**

Lavanya Sivakumar, Michael Schutte <michi@uiae.at>

**References**

J. Kim and T. Wilhelm. What is a complex graph? Physica A, 387:2637-2652, 2008

**Examples**

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:12), 0.5)

twoEdgesDeletedSubgraphComplexity(g)
```

---

variableZagreb	<i>Variable Zagreb index</i>
----------------	------------------------------

---

**Description**

This method calculates the variable Zagreb index.

**Usage**

```
variableZagreb(g, deg = NULL)
```

**Arguments**

**g** a graph as a graphNEL object.  
**deg** the degree of each node of g. Will be automatically calculated if not supplied.

**Value**

This method returns the variable Zagreb index of a graph as a double-precision floating point value.

**Author(s)**

Lavanya Sivakumar, Michael Schutte

**References**

S. Nikolic and G. Kovacevic and A. Milicevic and N. Trinajstic: The Zagreb Indices 30 Years After. Croatica Chemica Acta, 76:113-124, 2003

## Examples

```
library(graph)
set.seed(123)
g <- randomEGraph(as.character(1:8), 0.6)

# optional: pre-calculate degree of nodes in g
vec.degree <- graph::degree(g)

variableZagreb(g, vec.degree)
```

---

vertConnectMatrix      *Vertex Connectivity Matrix*

---

## Description

Calculates the vertex connectivity matrix.

## Usage

```
vertConnectMatrix(g)
```

## Arguments

*g*                      A graph as a graphNEL object.

## Details

for deatils see the vignette or the reference

## Value

VerCon\_Mat              Returns the vertex connectivity matrix.

## Author(s)

Lavanya Sivakumar

## References

Dehmer M, Sivakumar L, Varmuzua K: Uniquely Discriminating Molecular Structures Using Novel Eigenvalue Based Descriptors. *match* 2012, 67:147-172

## Examples

```
library(RBGL)
g <- randomGraph(1:8, 1:5, 0.36)
vertConnectMatrix(g)
```

---

vertexDegree	<i>Vertex degree equality-based information index.</i>
--------------	--

---

**Description**

This method calculates the vertex degree equality-based information index

**Usage**

```
vertexDegree(g, deg = NULL)
```

**Arguments**

g                    a graph as a graphNEL object.  
deg                  the degree of each node of g.

**Value**

This method returns the Vertex degree equality-based information index.

**Author(s)**

Laurin Mueller

**References**

D. Bonchev, Information Theoretic Indices for Characterization of Chemical Structures. Research Studies Press, Chichester, 1983.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

vertexDegree(g)
```

---

weightedID	<i>Weighted ID number</i>
------------	---------------------------

---

**Description**

This method calculates the weighted ID number.

**Usage**

```
weightedID(g, dsc=NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>dsc</code>	the distance-sum-connectivity matrix of <code>g</code> . Will be automatically calculated if not supplied.

**Value**

The result is a list of two floating point numbers, the weighted ID number (WID) and the self-returning ID number (SID). The former is based on the sum of all walks weighted according to vertex distance degrees. The latter is limited to self-returning walks.

**Author(s)**

Michael Schutte

**References**

K. Szymanski, W. Mueller, J. Knop, and N. Trinajstić. On the Identification Numbers for Chemical Structures. *International Journal of Quantum Chemistry*, 30(S20):173-183, 1986

**Examples**

```
set.seed(987)
g <- randomEGraph(LETTERS[1:10], 0.3)

weightedID(g)
```

---

`weightStrucFuncMatrix_exp`*Weighted Structure Function Matrix*

---

**Description**

Calculates the weighted structure function matrix with exponential weighting parameter `c_i`.

**Usage**

```
weightStrucFuncMatrix_exp(g)
```

**Arguments**

`g` A graph as a graphNEL object.

**Details**

for details see the vignette or the reference

**Value**

`weightStrucFuncMatrix_exp`  
Returns the weighted structure function matrix with exponential weighting parameter `c_i`.

**Author(s)**

Lavanya Sivakumar

**References**

Dehmer M, Sivakumar L, Varmuzua K: Uniquely Discriminating Molecular Structures Using Novel Eigenvalue Based Descriptors. *match* 2012, 67:147-172

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)
weightStrucFuncMatrix_exp(g)
```

---

`weightStrucFuncMatrix_lin`*Weighted Structure Function Matrix*

---

**Description**

Calculates the weighted structure function matrix with linear weighting parameter `c_i`.

**Usage**

```
weightStrucFuncMatrix_lin(g)
```

**Arguments**

`g` A graph as a graphNEL object.

**Details**

for details see the vignette or the reference

**Value**

`weightStrucFuncMatrix_lin`  
Returns the weighted structure function matrix with linear weighting parameter `c_i`.

**Author(s)**

Lavanya Sivakumar

**References**

Dehmer M, Sivakumar L, Varmuzua K: Uniquely Discriminating Molecular Structures Using Novel Eigenvalue Based Descriptors. *match* 2012, 67:147-172

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)
weightStrucFuncMatrix_lin(g)
```

---

wiener	<i>Wiener index</i>
--------	---------------------

---

**Description**

This method calculates the Wiener index.

**Usage**

```
wiener(g, dist = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>dist</code>	the distance matrix of the graph. If the parameter is empty the distance matrix will be calculated within the function.

**Value**

This method returns the Wiener index.

**Author(s)**

Laurin Mueller

**References**

H. Wiener, Structural Determination of Paraffin Boiling Points, Journal of the American Chemical Society, vol. 69, no. 1, pp. 17-20, Jan. 1947.

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

wiener(g)
```

---

zagreb1	<i>Zagreb group index 1</i>
---------	-----------------------------

---

**Description**

This method calculates the Zagreb group index 1.

**Usage**

```
zagreb1(g, deg = NULL)
```

**Arguments**

**g** a graph as a graphNEL object.  
**deg** the degree of each node of g.

**Value**

This method returns the Zagreb group index 1.

**Author(s)**

Laurin Mueller

**References**

M. V. Diudea, I. Gutman, and L. Jantschi, *Molecular Topology*. Nova Publishing, 2001, New York, NY, USA

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

zagreb1(g)
```

---

`zagreb2`*Zagreb group index 2*

---

**Description**

This method calculates the Zagreb group index 2.

**Usage**

```
zagreb2(g, deg = NULL)
```

**Arguments**

<code>g</code>	a graph as a graphNEL object.
<code>deg</code>	the degree of each node of <code>g</code> .

**Value**

This method returns the Zagreb group index 2.

**Author(s)**

Laurin Mueller

**References**

M. V. Diudea, I. Gutman, and L. Jantschi, *Molecular Topology*. Nova Publishing, 2001, New York, NY, USA

**Examples**

```
library(RBGL)
set.seed(123)
g <- randomGraph(1:8, 1:5, 0.36)

zagreb2(g)
```

# Index

## \*Topic **descriptors**

- atomBondConnectivity, [7](#)
- augmentedZagreb, [9](#)
- balabanID, [10](#)
- balabanJ, [11](#)
- balabanlike1, [12](#)
- balabanlike2, [13](#)
- bertz, [14](#)
- bonchev1, [15](#)
- bonchev2, [16](#)
- bondOrderID, [17](#)
- compactness, [21](#)
- complexityIndexB, [22](#)
- connectivityID, [23](#)
- degreeDistribution, [24](#)
- diameter, [25](#)
- distanceCodeCentric, [26](#)
- distanceDegreeCentric, [27](#)
- dobrynin, [30](#)
- efficiency, [32](#)
- eigenvalueBased, [33](#)
- energy, [35](#)
- estrada, [36](#)
- geometricArithmetic1, [38](#)
- geometricArithmetic2, [39](#)
- geometricArithmetic3, [40](#)
- globalClusteringCoeff, [42](#)
- graphIndexComplexity, [43](#)
- graphVertexComplexity, [44](#)
- harary, [45](#)
- huXuID, [46](#)
- hyperDistancePathIndex, [47](#)
- infoTheoreticGCM, [48](#)
- konstantinova, [49](#)
- laplacianEnergy, [51](#)
- laplacianEstrada, [52](#)
- localClusteringCoeff, [53](#)
- meanDistanceDeviation, [54](#)
- mediumArticulation, [55](#)

- minBalabanID, [56](#)
- minConnectivityID, [57](#)
- modifiedZagreb, [58](#)
- narumiKatayama, [59](#)
- normalizedEdgeComplexity, [60](#)
- offdiagonal, [61](#)
- primeID, [63](#)
- productOfRowSums, [64](#)
- radialCentric, [65](#)
- randic, [66](#)
- spanningTreeSensitivity, [68](#)
- spectralRadius, [69](#)
- topologicalInfoContent, [69](#)
- totalAdjacency, [70](#)
- twoEdgesDeletedSubgraphComplexity, [71](#)
- variableZagreb, [72](#)
- vertexDegree, [74](#)
- weightedID, [75](#)
- wiener, [78](#)
- zagreb1, [79](#)
- zagreb2, [80](#)

## \*Topic **graph matrices**

- adjacencyMatrix, [6](#)
- augmentedMatrix, [8](#)
- calculateDescriptors, [18](#)
- distanceMatrix, [28](#)
- distancePathMatrix, [28](#)
- distSumConnectMatrix, [29](#)
- extendedAdjacencyMatrix, [37](#)
- laplaceMatrix, [50](#)
- randomWalkMatrix, [67](#)
- vertConnectMatrix, [73](#)
- weightStrucFuncMatrix\_exp, [76](#)
- weightStrucFuncMatrix\_lin, [77](#)

## \*Topic **package**

- QuACN-package, [3](#)

## \*Topic **subgraphs**

- edgeDeletedSubgraphs, [32](#)

- getLargestSubgraph, [41](#)
- oneEdgeDeletedSubgraphComplexity, [62](#)
  
- adjacencyMatrix, [6](#)
- atomBondConnectivity, [7](#)
- augmentedMatrix, [8](#)
- augmentedZagreb, [9](#)
  
- balabanID, [10](#)
- balabanJ, [11](#)
- balabanlike1, [12](#)
- balabanlike2, [13](#)
- bertz, [14](#)
- bonchev1, [15](#)
- bonchev2, [16](#)
- bondOrderID, [17](#)
  
- calculateDescriptors, [18](#)
- compactness, [21](#)
- complexityIndexB, [22](#)
- connectivityID, [23](#)
  
- degreeDistribution, [24](#)
- diameter, [25](#)
- distanceCodeCentric, [26](#)
- distanceDegreeCentric, [27](#)
- distanceMatrix, [28](#)
- distancePathMatrix, [28](#)
- distSumConnectMatrix, [29](#)
- dobrynin, [30](#)
  
- edgeDeletedSubgraphs, [32](#)
- efficiency, [32](#)
- eigenvalueBased, [33](#)
- energy, [35](#)
- estrada, [36](#)
- extendedAdjacencyMatrix, [37](#)
  
- geometricArithmetic1, [38](#)
- geometricArithmetic2, [39](#)
- geometricArithmetic3, [40](#)
- getLabels, [41](#)
- getLargestSubgraph, [41](#)
- globalClusteringCoeff, [42](#)
- graphIndexComplexity, [43](#)
- graphVertexComplexity, [44](#)
  
- harary, [45](#)
- huXuID, [46](#)
  
- hyperDistancePathIndex, [47](#)
  
- infoTheoreticGCM, [48](#)
  
- konstantinova, [49](#)
  
- laplaceMatrix, [50](#)
- laplacianEnergy, [51](#)
- laplacianEstrada, [52](#)
- localClusteringCoeff, [53](#)
  
- meanDistanceDeviation, [54](#)
- mediumArticulation, [55](#)
- minBalabanID, [56](#)
- minConnectivityID, [57](#)
- modifiedZagreb, [58](#)
  
- narumiKatayama, [59](#)
- normalizedEdgeComplexity, [60](#)
  
- offdiagonal, [61](#)
- oneEdgeDeletedSubgraphComplexity, [62](#)
  
- primeID, [63](#)
- productOfRowSums, [64](#)
  
- QuACN (QuACN-package), [3](#)
- QuACN-package, [3](#)
  
- radialCentric, [65](#)
- randic, [66](#)
- randomWalkMatrix, [67](#)
  
- spanningTreeSensitivity, [68](#)
- spectralRadius, [69](#)
  
- topologicalInfoContent, [69](#)
- totalAdjacency, [70](#)
- twoEdgesDeletedSubgraphComplexity, [71](#)
  
- variableZagreb, [72](#)
- vertConnectMatrix, [73](#)
- vertexDegree, [74](#)
  
- weightedID, [75](#)
- weightStrucFuncMatrix\_exp, [76](#)
- weightStrucFuncMatrix\_lin, [77](#)
- wiener, [78](#)
  
- zagreb1, [79](#)
- zagreb2, [80](#)