QuACN: **Qu**antitative **A**nalyze of **C**omplex **N**etworks

Laurin AJ Mueller, Michael Schutte, Karl G Kugler, Matthias Dehmer[*]

October 18, 2011

Contents

1 Overview **1**
1.1 Installation . . . 1

2 Networks **2**
2.1 graphNEL . . . 2
2.2 Adjacency Matrix . . . 2
2.3 Extract the Largest Connected Subgraph . . . 3
2.4 Enumerate Edge-Deleted Subgraphs . . . 3

3 Network Descriptors **3**
3.1 Descriptors Based on Distances in a Graph . . . 4
3.2 Descriptors Based on Other Graph-Invariants . . . 7
3.3 Classical Entropy-based descriptors . . . 10
3.4 More recent Graph Complexity Measures . . . 13
3.5 Parametric Graph Entropy Measures . . . 16
3.6 Eigenvalue-based Descriptors . . . 18
3.7 Subgraph Measures . . . 23
3.8 ID numbers . . . 24

4 Calculating Multiple Descriptors at Once **26**

5 Session Info **27**

1 Overview

For information about the actual build see the projects website:

- R-Forge: `http://quacn.r-forge.r-project.org/`
- CRAN: `http://cran.r-project.org/web/packages/QuACN/`

This vignette provides an overview about the usage of `QuACN`.

Chapter 2 will describe how to import already exiting networks. In Chapter 3 a brief description of the implemented measures is presented, and it demonstrates how to call the related method in R.

1.1 Installation

`QuACN` uses the packages `graph` and `RBGL` from the *Bioconductor* project. Before installing `QuACN`, *Bioconductor* with the corresponding packages needs to be installed. For instructions see the *Bioconductor* website:

- Bioconductor: `http://www.bioconductor.org/`

Note, that `QuACN` also depends on the `Rmpfr` package. Therefore, the software GMP (`http://gmplib.org/`) and MPFR (`http://www.mpfr.org/`) needs to be installed to install the package correctly:

[*]Corresponding author.

- Windows: The package should install without problems.
- Ubuntu/Debian: Make sure that the libraries *libgmp3-dev* and *libmpfr-dev* are installed.

For more information see the corresponding websites, or the documentation of the `Rmpfr` package (`http://rmpfr.r-forge.r-project.org/`).

After installing *GMP* and *MPFR* everything is ready to install `QuACN`. Other dependencies will be installed automatically during the installation. To install the package from *CRAN* simply type:

```
> install.packages("QuACN")
```

2 Networks

This section shows how to create networks in R to use them with `QuACN`.

2.1 graphNEL

We generate a random graph with 8 nodes. This graph will be used to explain the implemented methods. To analyze a network the network has to be represented by a *graphNEL*-object, which is part of the Bioconductor `graph` package.

```
> library("QuACN")

Loading C code of R package 'Rmpfr': GMP using 64 bits per limb

> set.seed(666)
> g <- randomGraph(1:8, 1:5, 0.36)
> g

A graphNEL graph with undirected edges
Number of Nodes = 8
Number of Edges = 16
```

2.2 Adjacency Matrix

To create a *graphNEL* object from an adjacency matrix A, just type following command:

```
> A

  1 2 3 4 5 6 7 8
1 0 1 1 2 1 0 1 1
2 1 0 1 1 1 0 0 0
3 1 1 0 1 1 0 0 0
4 2 1 1 0 1 0 1 1
5 1 1 1 1 0 1 0 0
6 0 0 0 0 1 0 0 0
7 1 0 0 1 0 0 0 1
8 1 0 0 1 0 0 1 0

> g <- as(A, "graphNEL")
> g

A graphNEL graph with undirected edges
Number of Nodes = 8
Number of Edges = 16
```

Some descriptors, which are specially marked throughout this document, require vertex and/or edge weights. Known attributes are:

- `"atom"`: Atomic number of a graph vertex.

- "weight": Conventional bond order of an edge, i.e. 1 for single bonds, 2 for double bonds, 3 for triple bonds and 1.5 for aromatic bonds.

These can be set as follows:

```
> nodeDataDefaults(g, "atom") <- 6
> nodeData(g, "6", "atom") <- 8
> edgeDataDefaults(g, "weight") <- 1
> edgeData(g, "2", "3", "weight") <- 2
```

If existing networks are to be analyzed with QuACN, R offers several ways to import them. (It is important that the networks are represented by *graphNEL*-objects.) Note that there is no general procedure to get networks into an R workspace. Some possibilities to import network data are listed below:

- **Adjacency matrix**: A representation of a network as an adjacency matrix can be easily imported and converted into a *graphNEL* object.
- **Node- and Edge-List**: With a list of nodes and Edges it is easy to create a *graphNEL*-object.
- **read.graph()**: The read.graph() method of the *graph*-package offers the possibility to import graphs that a represented in different formats. For details see the manual of the *graph*-package.
- **System Biology Markup Language(SBML) [1]**: With the *RSBML*-package it is possible to import SBML-Models.
- **igrah-package**: Networks created with the *igraph*-package can be converted into graphNEL objects.

2.3 Extract the Largest Connected Subgraph

Many of the topological network descriptors that are implemented in QuACN only work on connected graphs. Often this is not the case with biological networks, so that the largest connected component (LCC) has to be extracted first. For extracting the LCC we provide the method getLargestSubgraph(g), as shown in [2]:

```
> g2 <- randomGraph(paste("A", 1:100, sep = ""), 1:4, p = 0.03)
> lcc <- getLargestSubgraph(g2)
> lcc
```

```
A graphNEL graph with undirected edges
Number of Nodes = 7
Number of Edges = 12
```

2.4 Enumerate Edge-Deleted Subgraphs

Some descriptors require a list of all distinct subgraphs which can be generated from a graph by removing one or two edges. The concerning methods obtain this information automatically, but for efficiency reasons, the user might want to pre-calculate and reuse it:

```
> sg.1ed <- edgeDeletedSubgraphs(g)
> sg.2ed <- edgeDeletedSubgraphs(sg.1ed)
```

Note that the method edgeDeletedSubgraphs(g) accepts lists or single instances of graphNEL objects or adjacency matrices, but it always returns a list of adjacency matrices.

3 Network Descriptors

This section provides a overview of the network descriptors that are included in the QuACN package. Here we describe the respective descriptor and how to call it in R.

Many descriptors have at least two parameters, the *graphNEL*-object and the distance matrix representing the network. It is not necessary to pass the distance matrix to a function. If the parameters stays empty or is set to *NULL* the distance matrix will be estimated within each function. But if the user wants to calculate more than one descriptor, it is recommended to calculate the distance matrix separately and pass it to each method. Some of the methods need the degree of each node or the adjacency matrix to calculate their results. If they were calculated once they should have kept for later use. For large networks in particular, it saves a lot of time to not calculate these parameters for each descriptor again, and will enhance the performance of the program to be developed.

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

In the definitions below, let $G = (N(G), E(G))$ be a finite and connected graph. $N(G)$ and $E(G)$ are called vertex and edge set of G, respectively. As $|N(G)| < \inf$, we can define $|N(G)| := N$.

3.1 Descriptors Based on Distances in a Graph

This section describes network measures based on distances in the network.

Wiener Index [3]:

$$W(G) := \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} d(v_i, v_j). \tag{1}$$

$d(v_i, v_j)$ stands for shortest distances between $v_i, v_j \in N(G)$.

```
> wien <- wiener(g)
> wiener(g, mat.dist)

[1] 44
```

Hararay Index [4]:

$$H(G) := \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} (d(v_i, v_j))^{-1}, \quad i \neq j. \tag{2}$$

```
> harary(g)

[1] 20.66667

> harary(g, mat.dist)

[1] 20.66667
```

Balaban J Index [5]:

$$J(G) := \frac{|E|}{\mu + 1} \sum_{(v_i,v_j)\in E} [DS_i DS_j]^{-\frac{1}{2}}, \tag{3}$$

```
> balabanJ(g)

[1] 2.207272

> balabanJ(g, mat.dist)

[1] 2.207272
```

where $|E(G)| := |E|$ denotes the number of edges of the complex network, DS_i denotes the distance sum (row sum) of v_i and $\mu := |E| + 1 - N$ denotes the cyclomatic number.

Mean distance deviation [6]:

See subsection 3.1.

Compactness [7]:

$$C(G) := \frac{4W}{N(N-1)}. \tag{4}$$

```
> compactness(g)
[1] 3.142857
> compactness(g, mat.dist)
[1] 3.142857
> compactness(g, mat.dist, wiener(g, mat.dist))
[1] 3.142857
```

Product of Row Sums Index [8]:

$$\mathrm{PRS}(G) = \prod_{i=1}^{N} \mu(v_i) \quad \text{or} \quad \log\big(\mathrm{PRS}(G)\big) = \log\left(\prod_{i=1}^{N} \mu(v_i)\right). \tag{5}$$

```
> productOfRowSums(g, log = FALSE)
[1] 190531440
> productOfRowSums(g, log = TRUE)
[1] 27.50545
> productOfRowSums(g, mat.dist, log = FALSE)
[1] 190531440
> productOfRowSums(g, mat.dist, log = TRUE)
[1] 27.50545
```

Hyper-distance-path Index [9]

$$D_P(G) := \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} d(v_i, v_j) + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \binom{d(v_i, v_j)}{2}. \tag{6}$$

```
> hyperDistancePathIndex(g)
[1] 62
> hyperDistancePathIndex(g, mat.dist)
[1] 62
> hyperDistancePathIndex(g, mat.dist, wiener(g, mat.dist))
[1] 62
```

Skorobogatov and Dobrynin [6]:

This method calculates several descriptors:

1. Vertex Eccentricity [6]:
$$e(v) := max_{u \in N(G)} d(u, v) \tag{7}$$

```
> dob <- dobrynin(g)
> dob <- dobrynin(g, mat.dist)
> dob$eccentricityVertex

1 2 3 4 5 6 7 8
2 2 2 2 2 3 3 3
```

2. Eccentricity of a graph [6]:
$$e(G) := \sum_{v \in N(G)} e(v) \tag{8}$$

```
> dob$eccentricityGraph

[1] 19
```

3. Average Vertex Eccentricity of a Graph [6]:
$$e_{av}(G) := \frac{e(G)}{N} \tag{9}$$

```
> dob$avgeccOfG

[1] 2.375
```

4. Vertex Eccentric [6]:
$$\Delta e(v) := |e(v) - e_{av}(G)| \tag{10}$$

```
> dob$ecentricVertex

    1     2     3     4     5     6     7     8
0.375 0.375 0.375 0.375 0.375 0.625 0.625 0.625
```

5. Eccentric of a Graph [6]:
$$\Delta G := \frac{1}{N} \sum_{v \in N(G)} \Delta e(v) \tag{11}$$

```
> dob$ecentricGraph

[1] 0.46875
```

6. Vertex Centrality [6]:
$$D(v) := \sum_{v \in N(G)} d(v, u) \tag{12}$$

```
> dob$vertexCentrality

 1  2  3  4  5  6  7  8
 9 11 11  9  9 15 12 12
```

7. Graph Integration [6]:
$$D(G) := \frac{1}{2} \sum_{v \in N(G)} D(v) \tag{13}$$

```
> dob$graphIntegration

[1] 44
```

8. Unipolarity [6]:
$$D^*(G) := min_{u \in N(G)} D(v) \tag{14}$$

```
> dob$unipolarity

[1] 9
```

9. Distance Vertex Deviation [6]:
$$\Delta D^*(v) := D(v) - D^*(G) \tag{15}$$

```
> dob$vertexDeviation

1 2 3 4 5 6 7 8
0 2 2 0 0 6 3 3
```

10. Variation of a Graph [6]:
$$var(g) := max_{u \in N(G)} \Delta D^*(v) \tag{16}$$

```
> dob$variation

[1] 6
```

11. Centralization [6]:
$$\Delta G^* := \sum_{v \in N(G)} \Delta D^*(v) \tag{17}$$

```
> dob$centralization

[1] 16
```

12. Average Distance of Graph Vertices [6]:
$$D_{av}(g) := \frac{2D(g)}{N} \tag{18}$$

```
> dob$avgDistance

[1] 11
```

13. Distance Vertex Deviation [6]:
$$\Delta D(v) := |D(v) - D_{av}(G)| \tag{19}$$

```
> dob$distVertexDeviation

1 2 3 4 5 6 7 8
2 0 0 2 2 4 1 1
```

14. Mean Distance Deviation [6]:
$$\Delta D(G) := \frac{1}{N} \sum_{v \in N(G)} \Delta D(v) \tag{20}$$

```
> dob$meanDistVertexDeviation

[1] 1.5
```

3.2 Descriptors Based on Other Graph-Invariants

This section describes network measures based on other invariants than distances.

Index of Total Adjacency [10]:

$$A(G) := \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}. \tag{21}$$

```
> totalAdjacency(g)
[1] 18
> totalAdjacency(g, mat.adj)
[1] 18
```

Zagreb Group Indices [11, 12]:

$$Z_1(G) := \sum_{i=1}^{N} k_{v_i}, \tag{22}$$

where k_{v_i} is the degree of the node v_i.

$$Z_2(G) := \sum_{(v_i,v_j)\in E} k_{v_i}k_{v_j}. \tag{23}$$

Modified, augmented and variable Zagreb indices:

$$\begin{aligned} MZI(G) &:= \sum_{(v_i,v_j)\in E} \frac{1}{k_{v_i}k_{v_j}}, & (24)\\ AZI(G) &:= \sum_{(v_i,v_j)\in E} \left(\frac{k_{v_i}k_{v_j}}{k_{v_i}+k_{v_j}-2}\right)^3, & (25)\\ VZI(G) &:= \sum_{(v_i,v_j)\in E} \frac{k_{v_i}+k_{v_j}-2}{k_{v_i}k_{v_j}}. & (26)\end{aligned}$$

```
> zagreb1(g)
[1] 32
> zagreb1(g, vec.degree)
[1] 32
> zagreb2(g)
[1] 282
> zagreb2(g, vec.degree)
[1] 282
> modifiedZagreb(g)
[1] 0.8666667
> modifiedZagreb(g, vec.degree)
[1] 0.8666667
> augmentedZagreb(g)
[1] 310.0767
> augmentedZagreb(g, vec.degree)
[1] 310.0767
> variableZagreb(g)
[1] 5.433333
> variableZagreb(g, vec.degree)
[1] 5.433333
```

Randić Connectivity Index [13]:

$$R(G) := \sum_{(v_i,v_j)\in E} [k_{v_i} k_{v_j}]^{-\frac{1}{2}}. \tag{27}$$

```
> randic(g)
[1] 3.352215
> randic(g, vec.degree)
[1] 3.352215
```

The Complexity Index B [10]:

$$B(G) := \sum_{i=1}^{N} \frac{k_{v_i}}{\mu(v_i)}. \tag{28}$$

```
> complexityIndexB(g)
[1] 3.182828
> complexityIndexB(g, mat.dist)
[1] 3.182828
> complexityIndexB(g, mat.dist, vec.degree)
[1] 3.182828
```

Normalized Edge Complexity [10]:

$$E_N(G) := \frac{A(G)}{N^2}. \tag{29}$$

```
> normalizedEdgeComplexity(g)
[1] 0.28125
> normalizedEdgeComplexity(g, totalAdjacency(g, mat.adj))
[1] 0.28125
```

Atom-bond Connectivity [14]:

$$ABC(G) := \sum_{(v_i,v_j)\in E} \sqrt{\frac{k_{v_i} + k_{v_j} - 2}{k_{v_i} k_{v_j}}} \tag{30}$$

```
> atomBondConnectivity(g)
[1] 8.643594
> atomBondConnectivity(g, vec.degree)
[1] 8.643594
```

Geometric-arithmetic Indices [15]:

$$\begin{aligned} GA1(G) &:= \sum_{(v_i,v_j)\in E} \frac{\sqrt{k_{v_i}k_{v_j}}}{\frac{1}{2}(k_{v_i}+k_{v_j})} && (31)\\ GA2(G) &:= \sum_{(v_i,v_j)\in E} \frac{\sqrt{n_i n_j}}{\frac{1}{2}(n_i+n_j)} && (32)\\ GA3(G) &:= \sum_{(v_i,v_j)\in E} \frac{\sqrt{m_i m_j}}{\frac{1}{2}(m_i+m_j)} && (33)\end{aligned}$$

where

$$\begin{aligned} n_i &:= |\{x \in N(G) : d(x,v_i) < d(x,v_j)\}|, && (34)\\ n_j &:= |\{x \in N(G) : d(x,v_j) < d(x,v_i)\}|, && (35)\\ m_i &:= |\{f \in E : d(f,v_i) < d(f,v_j)\}|, && (36)\\ m_j &:= |\{f \in E : d(f,v_j) < d(f,v_i)\}|. && (37)\end{aligned}$$

In this context, the distance between an edge $f = \{x, y\}$ and a vertex v is defined as $d(f, v) := \min\{d(x, v), d(y, v)\}$.

```
> geometricArithmetic1(g)

[1] 13.41511

> geometricArithmetic1(g, vec.degree)

[1] 13.41511

> geometricArithmetic2(g)

[1] 12.77876

> geometricArithmetic2(g, mat.dist)

[1] 12.77876

> geometricArithmetic3(g)

[1] 11.81318

> geometricArithmetic3(g, mat.dist)

[1] 11.81318
```

Narumi-Katayama Index [16]:

$$NK := \prod_{i=1}^{N} k_{v_i} \tag{38}$$

```
> narumiKatayama(g)

[1] 25920

> narumiKatayama(g, vec.degree)

[1] 25920
```

3.3 Classical Entropy-based descriptors

These measures are based on grouping the elements of an arbitrary graph invariant (vertices, edges, and distances etc.) using an equivalence criterion.

Topological Information Content [17, 18]:

$$I^V_{orb}(G) := -\sum_{i=1}^{k} \frac{|N_i^V|}{N} \log\left(\frac{|N_i^V|}{N}\right). \tag{39}$$

$|N_i^V|$ denotes the number of vertices belonging to the i-th vertex orbit.

```
> topologicalInfoContent(g)

$entropy
[1] 2.25

$orbits
[1] 2 2 1 1 2

> topologicalInfoContent(g, mat.dist)

$entropy
[1] 2.25

$orbits
[1] 2 2 1 1 2

> topologicalInfoContent(g, mat.dist, vec.degree)

$entropy
[1] 2.25

$orbits
[1] 2 2 1 1 2
```

Bonchev - Trinajstić Indices [19]:

$$I_D(G) := -\frac{1}{N}\log\left(\frac{1}{N}\right) - \sum_{i=1}^{\rho(G)} \frac{2k_i}{N^2}\log\left(\frac{2k_i}{N^2}\right), \tag{40}$$

$$I_D^W(G) := W(G)\log(W(G)) - \sum_{i=1}^{\rho(G)} ik_i \log(i). \tag{41}$$

$\rho(G)$ is the diameter of the graph (the maximum distance between two nodes). k_i is the occurrence of a distance possessing value i in the distance matrix of G.

```
> #I_D(G)
> bonchev1(g)

[1] 1.229843

> bonchev1(g, mat.dist)

[1] 1.229843

> #I^W_D(G)
> bonchev2(g)

[1] 173.1954

> bonchev2(g, mat.dist)

[1] 173.1954

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

[1] 173.1954
```

BERTZ Complexity Index [20]:

$$C(G) := 2N\log(N) - \sum_{i=1}^{k} |N_i| \log\left(|N_i|\right). \tag{42}$$

$|N_i|$ are the cardinalities of the vertex orbits as defined in Eqn. (39).

```
> bertz(g)

[1] 42

> bertz(g, mat.dist)

[1] 42

> bertz(g, mat.dist, vec.degree)

[1] 42
```

Radial Centric Information Index [21]:

$$I_{C,R}(G) := \sum_{i=1}^{k} \frac{|N_i^e|}{N} \log\left(\frac{|N_i^e|}{N}\right). \tag{43}$$

$|N_i^e|$ is the number of vertices having the same eccentricity.

```
> radialCentric(g)

[1] 0.954434

> radialCentric(g, mat.dist)

[1] 0.954434
```

Vertex Degree Equality-based Information Index [21]:

$$I_{deg}(G) := \sum_{i=1}^{\bar{k}} \frac{|N_i^{k_v}|}{N} \log\left(\frac{|N_i^{k_v}|}{N}\right). \tag{44}$$

$|N_i^{k_v}|$ is the number of vertices with degree equal to i and $\bar{k} := \max_{v \in N(G)} k_v$.

```
> vertexDegree(g)

[1] 2.25

> vertexDegree(g, vec.degree)

[1] 2.25
```

Balaban-like Information Indices [22]:

Note that this class of Descriptors return *Inf* for graphs with $N < 3$.

$$U(G) := \frac{|E|}{\mu + 1} \sum_{(v_i,v_j)\in E} [u(v_i)u(v_j)]^{-\frac{1}{2}}, \tag{45}$$

$$X(G) := \frac{|E|}{\mu + 1} \sum_{(v_i,v_j)\in E} [x(v_i)x(v_j)]^{-\frac{1}{2}}, \tag{46}$$

where

$$
\begin{aligned}
u(v_i) &:= -\sum_{j=1}^{\sigma(v_i)} \frac{j|S_j(v_i,G)|}{\mu(v_i)} \log\left(\frac{j}{\mu(v_i)}\right), & (47)\\
x(v_i) &:= -\mu(v_i)\log(d(v_i)) - y_i, & (48)\\
y_i &:= \sum_{j=1}^{\sigma(v_i)} j|S_j(v_i,G)|\log(j), & (49)\\
\mu(v_i) &:= \sum_{j=1}^{N} d(v_i,v_j) = \sum_{j=1}^{N} j|S_j(v_i,G)|. & (50)
\end{aligned}
$$

```
> #Balaban-like information index U(G)
> balabanlike1(g)
```

```
[1] 8.236938
```

```
> balabanlike1(g, mat.dist)
```

```
[1] 8.236938
```

```
> #Balaban-like information index X(G)
> balabanlike2(g)
```

```
[1] 0.7589271
```

```
> balabanlike2(g, mat.dist)
```

```
[1] 0.7589271
```

Graph Vertex Complexity Index [23]:

$$I_V(G) := \sum_{i=1}^{N} v_i^c, \tag{51}$$

where v_i^c is the so-called vertex complexity expressed by

$$v_i^c := \sum_{j=0}^{\sigma(v_i)} \frac{k_j^{v_i}}{N} \log\left(\frac{k_j^{v_i}}{N}\right). \tag{52}$$

$k_k^{v_i}$ is the number of distances starting from $V_i \in N(G)$ equal to j.

```
> graphVertexComplexity(g)
```

```
[1] -12.08022
```

```
> graphVertexComplexity(g, mat.dist)
```

```
[1] -12.08022
```

3.4 More recent Graph Complexity Measures

Medium Articulation [24]:

$$MAg(G) := MA_R(G) \cdot MA_I(G) \tag{53}$$

with the redundancy

$$
\begin{aligned}
MA_R(G) &:= 4\left(\frac{R(G)-R_{\text{path}}(G)}{R_{\text{clique}}(G)-R_{\text{path}}(G)}\right)\left(1-\frac{R(G)-R_{\text{path}}(G)}{R_{\text{clique}}(G)-R_{\text{path}}(G)}\right) & (54)\\
R(G) &:= \frac{1}{m}\sum_{i,j>i} \log(d_i d_j) & (55)\\
R_{\text{clique}}(G) &= 2\log(N-1) & (56)\\
R_{\text{path}}(G) &= 2\frac{N-2}{N-1}\log 2 & (57)
\end{aligned}
$$

and the mutual information

$$MA_I(G) := 4\left(\frac{I(G) - I_{\text{clique}}(G)}{I_{\text{path}}(G) - I_{\text{clique}}(G)}\right)\left(1 - \frac{I(G) - I_{\text{clique}}(G)}{I_{\text{path}}(G) - I_{\text{clique}}(G)}\right) \quad (58)$$

$$I(G) := \frac{1}{m}\sum_{i,j>i} \log\left(\frac{2m}{d_i d_j}\right) \quad (59)$$

$$I_{\text{clique}}(G) = \log\frac{N}{N-1} \quad (60)$$

$$I_{\text{path}}(G) = \log(N-1) - \frac{N-3}{N-1}\log 2 \quad (61)$$

```
> mediumArticulation(g)
```

```
[1] 0.7722091
```

Efficiency Complexity [24]:

$$\text{Ce(G)} := 4\left(\frac{E(G) - E_{\text{path}}(G)}{1 - E_{\text{path}}(G)}\right)\left(1 - \frac{E(G) - E_{\text{path}}(G)}{1 - E_{\text{path}}(G)}\right) \quad (62)$$

$$E(G) := \frac{2}{N(N-1)}\sum_i \sum_{j>i} \frac{1}{d_{ij}} \quad (63)$$

$$E_{\text{path}}(G) = \frac{2}{N(N-1)}\sum_{i=1} N - 1\frac{N-i}{i} \quad (64)$$

```
> efficiency(g)
```

```
[1] 0.999175
```

```
> efficiency(g, mat.dist)
```

```
[1] 0.999175
```

Graph Index Complexity [24]:

$$Cr(G) := 4c_r(1 - c_r) \quad (65)$$

where $c_r = \frac{r - 2\cos\frac{\pi}{N+1}}{N-1-2\cos\frac{\pi}{N+1}}$ and r is the largest eigenvalue of the adjacency matrix of the graph.

```
> graphIndexComplexity(g)
```

```
[1] 0.8886164
```

Offdiagonal complexity [24]:

$$OdC(G) := -\frac{1}{log(N-1)}\sum_{n=0}^{k_{\max}-1} \tilde{a}_n \log \tilde{a}_n, \quad (66)$$

with $\tilde{a}_n = \frac{a_n}{\sum_{m=0}^{k_{\max}-1} a_m}$ and $a_n = \sum_{i=1}^{k_{\max}-N} c_{i,i+N}$, where $k_{\max}$ is the maximum degree of all nodes in the graph, and c_{ij} is the number of all neighbors with degree $j \geq i$ of all nodes with degree i.

```
> offdiagonal(g)
```

```
[1] 0.772423
```

```
> offdiagonal(g, vec.degree)
```

```
[1] 0.772423
```

Spanning Tree Sensitivity [24]:

$$STS(G) := \frac{-\sum_l a_l \log a_l}{\log m_{cu}}, \tag{67}$$

with $m_{cu} = n^{1.68} - 10$, $a_l = \frac{S^l_{ij}}{\sum_r^k S^r_{ij}}$, $S_{ij} = s_{ij} - (\min\{s_{ij}\} - 1)$ and $\{S^1_{ij}, S^2_{ij}, \ldots, S^k_{ij}\}$ being an ordered list of all k different S_{ij}. s_{ij} is the number of spanning trees in the graph minus the number of spanning trees of the subgraph with the edge $\{v_i, v_j\}$ deleted. Analogously, the spanning tree sensitivity differences measure is defined as

$$STSD(G) := \frac{-\sum_l b_l \log b_l}{\log m_{cu}}, \tag{68}$$

with $b_l = \frac{Ld_l}{\sum_r^d Ld_r}$, where $\{Ld_1, Ld_2, \ldots, Ld_d\}$ is the ordered list of all unique differences $S^m_{ij} - S^{m-1}_{ij}$.

```
> spanningTreeSensitivity(g)

$STS
[1] 0.4033211

$STSD
[1] 0.2846556

> spanningTreeSensitivity(g, sg.1ed)

$STS
[1] 0.4033211

$STSD
[1] 0.2846556
```

Distance Degree/Code Centric Indices [25]:

$$I_{\mathrm{C,deg}}(G) := -\sum_{i=1}^{D} \frac{d_i}{N} \log_2 \frac{d_i}{N}, \tag{69}$$

$$I_{\mathrm{C,code}}(G) := -\sum_{i=1}^{C} \frac{c_i}{N} \log_2 \frac{c_i}{N}. \tag{70}$$

where d_i is the number of vertices with the same eccentricity and the same vertex distance degree (i. e., equal row sums in the distance matrix), and c_i is the number of vertices with the same vertex distance code (i. e., the same numbers in their rows in the distance matrix). D and C are the respective numbers of equivalence classes.

```
> distanceDegreeCentric(g)

[1] 1.905639

> distanceDegreeCentric(g, mat.dist)

[1] 1.905639

> distanceCodeCentric(g)

[1] 1.905639

> distanceCodeCentric(g, mat.dist)

[1] 1.905639
```

3.5 Parametric Graph Entropy Measures

Measures of this group [26, 27] 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.

$$I_f(G) := -\sum_{i=1}^{N} \frac{f(v_i)}{\sum_{j=1}^{N} f(v_j)} \log\left(\frac{f(v_i)}{\sum_{j=1}^{N} f(v_j)}\right), \tag{71}$$

where $I_f(G)$ represents a family of graph entropy [26] measures depending on the information functional. Further we implemented the following measurement[27]:

$$I_f^{\lambda}(G) := \lambda\left(\log(N) + \sum_{i=1}^{N} p(v_i)\log(p(v_i))\right), \tag{72}$$

$$p(v_i) := \frac{f(v_i)}{\sum_{j=1}^{N} f(v_j)}, \tag{73}$$

where $p^V(v_i)$ are the vertex probabilities, and $\lambda > 0$ is a scaling constant. This measure can be interpreted as the distance between the entropy defined in equation 71 and maximum entropy ($\log(N)$). We integrated 4 different information functionals [26, 28]:

1. An information functional using the j-spheres (*"sphere"*):

$$f^V(v_i) := c_1|S_1(v_i, G)| + c_2|S_2(v_i, G)| + \cdots + c_{\rho(G)}|S_{\rho(G)}(v_i, G)|, \tag{74}$$

   where $c_k > 0$.

2. An information functional using path lengths (*"pathlength"*):

$$f^P(v_i) := c_1 l(P(L_G(v_i, 1))) + c_2 l(P(L_G(v_i, 2))) + \cdots + c_{\rho(G)} l(P(L_G(v_i, \rho(G)))), \tag{75}$$

   where $c_k > 0$.

3. An information functional using vertex centrality(*"vertcent"*) :

$$f^C(v_i) := c_1\beta^{L_G(v_i,1)}(v_i) + c_2\beta^{L_G(v_i,2)}(v_i) + \cdots + c_{\rho(G)}\beta^{L_G(v_i,\rho(G))}(v_i), \tag{76}$$

   where $c_k > 0$.

4. Calculates the degree-degree association index(*"degree"*) [28]:

$$f^{\Delta}(v_i) := \alpha^{c_1\Delta^G(v_i,1) + c_2\Delta^G(v_i,2) + \cdots + c_{\rho(G)}\Delta^G(v_i,\rho(G))}, \tag{77}$$

   where $c_k > 0$, $1 \le k \le \rho(G)$ and $\alpha > 0$. Note that f^{Δ} is well-defined for $\alpha > 0$. Please consider that the results of the degree-degree association index are often very close to zero and can only be represented with a special data type (see the hint at the end of this section).

We implemented 4 different settings (as example settings) for weighting the parameters c_i ($\rho(G)$ represents the diameter of the network):

1. constant

$$c_1 := 1,\ c_2 := 1, \cdots, c_{\rho(G)} := 1. \tag{78}$$

2. linear

$$c_1 := \rho(G),\ c_2 := \rho(G) - 1, \cdots, c_{\rho(G)} := 1. \tag{79}$$

3. quadratic

$$c_1 := \rho(G)^2, c_2 := (\rho(G) - 1)^2, \cdots, c_{\rho(G)} := 1. \tag{80}$$

4. exponential

$$c_1 := \rho(G), c_2 := \rho(G)e^1, \cdots, c_{\rho(G)} := \rho(G)e^{-\rho(G)+1}. \tag{81}$$

To call this type of network measure we provide the method *infoTheoreticGCM*. It has following input parameters:

- *g*: the network as a graphNEL object - it is the only mandatory parameter
- *dist*: the distance matrix of g
- coeff: specifies the weighting parameter: "const", "lin", "quad", "exp", "const" or "cust" are available constants. If it is set to "cust", a customized weighting schema has to be specified through the *custCoeff* parameter.
- *infofunct*: specifies the information functional: "sphere", "pathlength", "vertcent" or "degree" are available settings.
- *lambda*: scaling constant for the distance, default set to 1000.
- *custCoeff*: specifies the customized weighting schema. *coeff* must be set to "const" in order to use it.
- *alpha*: alpha for degree degree association.
- *prec*: 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.

Note that some combinations of these settings can cause the descriptor to return *NaN*. In that case it is the user's responsibility to check for *warnings*. For `infofunct="degree"` in particular, also see the note below.

The method returns a list with following entries:

- *entropy*: contains the entropy, see formula 71
- *distance*: contains the distance described in formula 72
- *pis*: contains the probability distribution, see formula 73
- *fvi*: contains the values of the used information functional for each vertex v_i

```
> l1 <- infoTheoreticGCM(g)
> l2 <- infoTheoreticGCM(g, mat.dist, coeff = "lin", infofunct = "sphere",
+     lambda = 1000)
> l3 <- infoTheoreticGCM(g, mat.dist, coeff = "const", infofunct = "pathlength",
+     lambda = 4000)
> l4 <- infoTheoreticGCM(g, mat.dist, coeff = "quad", infofunct = "vertcent",
+     lambda = 1000)
> l5 <- infoTheoreticGCM(g, mat.dist, coeff = "exp", infofunct = "degree",
+     lambda = 1000)
> l1

$entropy
[1] 2.990321

$distance
[1] 9.679226

$pis
         1          2          3          4          5          6          7 
0.13970588 0.12500000 0.12500000 0.13970588 0.13970588 0.09558824 0.11764706 
         8 
0.11764706 

$fvis
 1  2  3  4  5  6  7  8 
19 17 17 19 19 13 16 16 

> l5
```

```
$entropy
[1] 1.546569

$distance
[1] 1453.431

$pis
           1            2            3            4            5            6 
4.474312e-01 2.658896e-02 2.658896e-02 4.474312e-01 5.075579e-02 1.196450e-03 
           7            8 
3.710288e-06 3.710288e-06 

$fvis
           1            2            3            4            5            6 
2.540206e-12 1.509538e-13 1.509538e-13 2.540206e-12 2.881564e-13 6.792618e-15 
           7            8 
2.106446e-17 2.106446e-17 
```

Important: Note, the functional based on degree-degree associations (`infofunct="degree"`) can result in values that cannot be represented by standard data types. This problem manifests itself in *NaN* as return values. Note, that this issue can be avoided by specifying a floating-point precision value greater than 53, using the parameter *prec* (e.g. `prec=128` is usually enough). In this case, the `Rmpfr` package will be used and the list, returned by the function will contain vectors of the class *mpfr*. These vectors can be used as usual numeric vectors, except that all calculations will result in *mpfr* vectors. Note, that `as.double` can be used to convert such a vector back to the regular *numeric* vector once the result is in the representable range (between 10^{-380} and 10^{380}). The following example shows how to work with vectors of type *mpfr*:

```
> l5mpfr <- infoTheoreticGCM(g, mat.dist, coeff = "exp", infofunct = "degree",
+     lambda = 1000, prec = 128)
> l5mpfr$entropy

1 'mpfr' number of precision  128   bits 
[1] 1.546568792292280720674101903940257498633

> l5mpfr$entropy * 2^3

1 'mpfr' number of precision  128   bits 
[1] 12.37255033833824576539281523152205998907

> as.double(l5mpfr$entropy * 2^3)

[1] 12.37255
```

For more details about *mpfr* vectors, please consult the `Rmpfr` documentation.

3.6 Eigenvalue-based Descriptors

This class contains eigenvalue-based Descriptors proposed in Dehmer et. al [28].

$$H_{M_s}(G) = \sum_{i=1}^{k} \frac{|\lambda_i|^{\frac{1}{s}}}{\sum_{j=1}^{k} |\lambda_j|^{\frac{1}{s}}} \log\left(\frac{|\lambda_i|^{\frac{1}{s}}}{\sum_{j=1}^{k} |\lambda_j|^{\frac{1}{s}}}\right), \tag{82}$$

$$S_{M_s}(G) = |\lambda_1|^{\frac{1}{s}} + |\lambda_2|^{\frac{1}{s}} + \ldots + |\lambda_k|^{\frac{1}{s}}, \tag{83}$$

$$IS_{M_s}(G) = \frac{1}{|\lambda_1|^{\frac{1}{s}} + |\lambda_2|^{\frac{1}{s}} + \ldots + |\lambda_k|^{\frac{1}{s}}}, \tag{84}$$

$$P_{M_s}(G) = |\lambda_1|^{\frac{1}{s}} \cdot |\lambda_2|^{\frac{1}{s}} \ldots |\lambda_k|^{\frac{1}{s}}, \tag{85}$$

$$IP_{M_s}(G) = \frac{1}{|\lambda_1|^{\frac{1}{s}} \cdot |\lambda_2|^{\frac{1}{s}} \ldots |\lambda_k|^{\frac{1}{s}}}, \tag{86}$$

Using this function, it is possible to calculate 5 descriptors ($H_{M_s(G)}$, $S_{M_s(G)}$, $IS_{M_s(G)}$, $P_{M_s(G)}$, $IP_{M_s(G)}$) for 10 different matrices:

1. Adjacency matrix

```
> eigenvalueBased(g, adjacencyMatrix, 2)

$HMs
[1] 2.924559

$SMs
[1] 10.45478

$ISMs
[1] 0.09565

$PMs
[1] 5.656854

$IPMs
[1] 0.1767767
```

2. Laplacian matrix

```
> eigenvalueBased(g, laplaceMatrix, 2)

$HMs
[1] 2.728232

$SMs
[1] 15.14344

$ISMs
[1] 0.06603521

$PMs
[1] 2.335730e-06

$IPMs
[1] 428131.6
```

3. Distance matrix

```
> eigenvalueBased(g, distanceMatrix, 2)

$HMs
[1] 2.812274

$SMs
[1] 12.07332

$ISMs
[1] 0.08282723

$PMs
[1] 9.797959

$IPMs
[1] 0.1020621
```

4. Distance path Matrix

```
> eigenvalueBased(g, distancePathMatrix, 2)

$HMs
[1] 2.770978

$SMs
[1] 14.89195

$ISMs
[1] 0.06715037

$PMs
[1] 36.61967

$IPMs
[1] 0.02730773
```

5. Augmented vertex degree matrix

```
> eigenvalueBased(g, augmentedMatrix, 2)

$HMs
[1] 2.798655

$SMs
[1] 13.96496

$ISMs
[1] 0.0716078

$PMs
[1] 28.48828

$IPMs
[1] 0.03510216
```

6. Extended adjacency matrix

```
> eigenvalueBased(g, extendedAdjacencyMatrix, 2)

$HMs
[1] 2.926072

$SMs
[1] 10.94290

$ISMs
[1] 0.0913835

$PMs
[1] 8.199051

$IPMs
[1] 0.1219653
```

7. Vertex Connectivity matrix

```
> eigenvalueBased(g, vertConnectMatrix, 2)
```

```
$HMs
[1] 2.942791

$SMs
[1] 4.976892

$ISMs
[1] 0.2009286

$PMs
[1] 0.01643355

$IPMs
[1] 60.85111
```

8. Random Walk Markov matrix

```
> eigenvalueBased(g, randomWalkMatrix, 2)

$HMs
[1] 2.942791

$SMs
[1] 4.976892

$ISMs
[1] 0.2009286

$PMs
[1] 0.01643355

$IPMs
[1] 60.85111
```

9. Weighted structure function matrix IM_1

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

$HMs
[1] 0.8690336

$SMs
[1] 3.293587

$ISMs
[1] 0.3036204

$PMs
[1] 1.289736e-29

$IPMs
[1] 7.753523e+28
```

10. Weighted structure function matrix IM_2

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

$HMs
[1] 1.038954
```

```
$SMs
[1] 3.450187

$ISMs
[1] 0.2898393

$PMs
[1] 9.348347e-36

$IPMs
[1] 1.069708e+35
```

For a detailed description of this class see Dehmer et. al [28].

Graph Energy and Laplacian Energy [29]:

$$E(G) := \sum_{i=1}^{N} |\lambda_i| \tag{87}$$

$$LE(G) := \sum_{i=1}^{N} \left|\mu_i - \frac{2m}{n}\right| \tag{88}$$

where λ_k are the eigenvalues of the adjacency matrix and μ_k those of the Laplacian matrix of the graph.

```
> energy(g)

[1] 15.19639

> laplacianEnergy(g)

[1] 21.86179
```

Estrada [30] and Laplacian Estrada [31] Indices:

$$EE(G) := \sum_{i=1}^{N} e^{\lambda_i} \tag{89}$$

$$LEE(G) := \sum_{i=1}^{N} e^{\mu_i} \tag{90}$$

with λ_k and μ_k defined as above.

```
> estrada(g)

[1] 207.9575

> laplacianEstrada(g)

[1] 10832.26
```

Spectral Radius:

$$SpRad(G) := \max_i \{|\lambda_i|\} \tag{91}$$

```
> spectralRadius(g)

[1] 5.294174
```

3.7 Subgraph Measures

One-edge-deleted Subgraph Complexity [24]:

$$C_{1e,ST}(G) := \frac{N_{1e,ST} - 1}{m_{cu} - 1} \tag{92}$$

$$C_{1e,Spec}(G) := \frac{N_{1e,Spec} - 1}{m_{cu} - 1} \tag{93}$$

$N_{1e,ST}$ is the number of one-edge-deleted subgraphs which are different with regard to the number of spanning trees. Similarly, $N_{1e,Spec}$ is the number of one-edge-deleted subgraphs which are different with regard to spectra of the Laplacian and signless Laplacian matrix. m_{cu} is defined as $n^{1.68} - 10$.

```
> oneEdgeDeletedSubgraphComplexity(g)

$C_1eST
[1] 0.3196399

$C_1eSpec
[1] 0.593617

> oneEdgeDeletedSubgraphComplexity(g, sg.1ed)

$C_1eST
[1] 0.3196399

$C_1eSpec
[1] 0.593617
```

Two-edges-deleted Subgraph Complexity [24]:

$$C_{2e,Spec}(G) := \frac{N_{2e,Spec} - 1}{\binom{m_{cu}}{2} - 1} \tag{94}$$

where m_{cu} is defined like above and $N_{2e,Spec}$ is the number of two-edges-deleted subgraphs which are different with regard to spectra of the Laplacian and signless Laplacian matrix.

```
> twoEdgesDeletedSubgraphComplexity(g)

[1] 0.3603647

> twoEdgesDeletedSubgraphComplexity(g, sg.2ed)

[1] 0.3603647
```

Local Clustering Coefficient [32, 33]:

Let $G_{N(v)} = (V_{N(v)}, E_{N(v)})$ be the subgraph of G that contains all neighborhood vertices and their edges. Then the local clustering coefficient of a graph G is defined by

$$C_v(G) := \frac{E_{N(v)}}{\frac{V_{N(v)} * (V_{N(v)} - 1)}{2}}. \tag{95}$$

```
> localClusteringCoeff(g)

  1   2   3   4   5   6   7   8
0.6 1.0 1.0 0.6 0.6 0.0 1.0 1.0

> localClusteringCoeff(g, deg = vec.degree)

  1   2   3   4   5   6   7   8
0.6 1.0 1.0 0.6 0.6 0.0 1.0 1.0
```

Global Clustering Coefficient [32, 33]:

$$C(G) := \sum_{v \in N(g)} := \frac{1}{N} * C_v \tag{96}$$

```
> loccc <- localClusteringCoeff(g)
> globalClusteringCoeff(g)
[1] 0.725
> globalClusteringCoeff(g, loc = loccc)
[1] 0.725
```

3.8 ID numbers

Randić Connectivity ID Number [34]:

$$CID := N + \sum_{^m p_{ij}} w_{ij}, \tag{97}$$

where $^m p_{ij}$ are all paths of length $m > 0$, and w_{ij} is a path weight defined as

$$w_{ij} = \prod_{b=1}^{m} \left(\delta_{b(1)}\delta_{b(2)}\right)_b^{-1/2}, \tag{98}$$

with the sum running over all edges in the path and $b(1)$, $b(2)$ referring to the two vertices incident to the bth edge.

```
> connectivityID(g)
[1] 17.53585
> connectivityID(g, deg = vec.degree)
[1] 17.53585
```

MINCID [35]:

$$MINCID := N + \sum_{^{\min} p_{ij}} w_{ij}, \tag{99}$$

where the sum runs over all shortest paths $^{\min}p_{ij}$ between the vertices v_i and v_j, and w_{ij} is taken from equation 98.

```
> minConnectivityID(g)
[1] 12.76619
> minConnectivityID(g, deg = vec.degree)
[1] 12.76619
```

Prime ID Number [36]:

$$PID := N + \sum_{^m p_{ij}} w_{ij}, \tag{100}$$

with $^m p_{ij}$ like above and the path weight w_{ij}

$$w_{ij} = \prod_{b=1}^{m} pn_b^{-1/2}, \tag{101}$$

where pn_b is a prime number chosen according to the degrees of the vertices adjacent to the bth edge.

```
> primeID(g)
[1] 16.08181
> primeID(g, deg = vec.degree)
[1] 16.08181
```

Conventional Bond Order ID Number [37]:

$$\pi ID := N + \sum_{{}^m p_{ij}} w_{ij}, \tag{102}$$

with ${}^m p_{ij}$ like above and the path weight w_{ij}

$$w_{ij} = \prod_{b=1}^{m} \pi_b^*, \tag{103}$$

where π_b^* is the conventional bond order of the bth edge.

The conventional bond order must be set as the `"weight"` edge data attribute of the input graph.

```
> bondOrderID(g)
```

```
[1] 2048
```

Balaban ID Number [38]:

$$BID := N + \sum_{{}^m p_{ij}} w_{ij}, \tag{104}$$

with ${}^m p_{ij}$ like above and the path weight w_{ij}

$$w_{ij} = \prod_{b=1}^{m} \left(\sigma_{b(1)} \cdot \sigma_{b(2)}\right)_b^{-1/2}, \tag{105}$$

where σ_k is the vertex distance degree and $b(1), b(2)$ refer to the vertices adjacent to the edge b.

```
> balabanID(g)
```

```
[1] 10.36261
```

```
> balabanID(g, dist = mat.dist)
```

```
[1] 10.36261
```

MINBID [35]:

$$MINBID := N + \sum_{{}^{\min} p_{ij}} w_{ij}, \tag{106}$$

where the sum runs over all shortest paths ${}^{\min}p_{ij}$ between the vertices v_i and v_j, and w_{ij} is taken from equation 105.

```
> minBalabanID(g)
```

```
[1] 9.741806
```

```
> minBalabanID(g, dist = mat.dist)
```

```
[1] 9.741806
```

Weighted ID Number [39]:

$$WID := N - \frac{1}{N} + \frac{ID^*}{N^2}, \tag{107}$$

with

$$ID^* := \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij}^*, \tag{108}$$

$$W^* := \sum_{k=0}^{N-1} {}^{\sigma}\chi^k, \tag{109}$$

where ${}^{\sigma}\chi$ is the distance-sum-connectivity matrix.

```
> weightedID(g)

$WID
[1] 8.091842

$SID
[1] 8.482815
```

Hu-Xu ID Number [40]:

$$HXID := \sum_{i=1}^{N} AID_i^2 \tag{110}$$

with

$$\begin{aligned} AID_i &:= \sum_{j=1}^{N} w_{ij}, & (111)\\ w_{ij} &:= \prod_{a=2}^{m+1} \left(\frac{\pi^*_{a-1,a}}{\cdot} \frac{1}{\delta'_{a-1} \cdot \delta'_a} \right)^{1/2}, & (112)\\ \delta'_a &:= \delta_a \cdot \sqrt{Z_a}, & (113) \end{aligned}$$

where Z_a is the atomic number of v_a.

The huXuID method requires the input graph to store the atomic numbers in the "atom" vertex data attribute and the conventional bond order in the "weight" edge data attribute.

```
> huXuID(g)

[1] 1.014972

> huXuID(g, deg = vec.degree)

[1] 1.014972
```

4 Calculating Multiple Descriptors at Once

The calculateDescriptors function provides a simple interface to calculate a set of descriptors on a list of input graphs. The result is returned as a data frame. The desired functions can be specified by name or by number. It is also possible to name the columns according to the names given in this document.

Please see the function documentation for a detailed description and a full list of the supported descriptors together with their numbers.

```
> calculateDescriptors(g, "wiener")

  wiener
1     44

> calculateDescriptors(g, 1001)

  wiener
1     44

> calculateDescriptors(g, 2000, labels = TRUE)

   A Z[1] Z[2]       MZI      AZI      VZI        R        B    E[N]      ABC
1 18   32  282 0.8666667 310.0767 5.433333 3.352215 3.182828 0.28125 8.643594
       GA1      GA2      GA3    NK
1 13.41511 12.77876 11.81318 25920
```

5 Session Info

```
> sessionInfo()

R version 2.12.1 (2010-12-16)
Platform: x86_64-pc-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=C              LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] QuACN_1.3.4    Rmpfr_0.4-3    combinat_0.0-8 igraph_0.5.5-2 RBGL_1.28.0
[6] graph_1.28.0

loaded via a namespace (and not attached):
[1] tools_2.12.1
```

References

[1] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. L. Novère, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, J. Wang, and S. B. M. L. Forum, "The Systems Biology Markup Language (SBML): A Medium for Representation and Exchange of Biochemical Network Models." *Bioinformatics*, vol. 19, no. 4, pp. 524–531, Mar 2003.

[2] F. Hahne, W. Huber, R. Gentleman, and S. Falcon, *Bioconductor Case Studies*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[3] H. Wiener, "Structural Determination of Paraffin Boiling Points," *Journal of the American Chemical Society*, vol. 69, no. 1, pp. 17–20, Jan. 1947. [Online]. Available: http://dx.doi.org/10.1021/ja01193a005

[4] 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.

[5] A. T. Balaban, "Highly Discriminating Distance-based Topological Index," *Chem.Phys.Lett.*, vol. 89, pp. 399–404, 1982.

[6] V. A. Skorobogatov and A. A. Dobrynin, "Metrical Analysis of Graphs," *Commun. Math. Comp. Chem.*, vol. 23, pp. 105–155, 1988.

[7] J. K. Doyle and J. E. Garver, "Mean Distance in a Graph," *Discrete Mathematics*, vol. 17, pp. 147–154, 1977.

[8] 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.

[9] R. Todeschini, V. Consonni, and R. Mannhold, *Handbook of Molecular Descriptors.* Wiley-VCH, 2002, weinheim, Germany.

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

[11] M. V. Diudea, I. Gutman, and L. Jantschi, *Molecular Topology.* Nova Publishing, 2001, new York, NY, USA.

[12] S. Nikolić, G. Kovačević, A. Milicević, and N. Trinajstić, "The Zagreb Indices 30 Years After," *Croatica Chemica Acta*, vol. 76, pp. 113–124, 2003.

[13] 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.

[14] E. Estrada, L. Torres, L. Rodríguez, and I. Gutman, "An Atom-Bond Connectivity Index: Modelling the Enthalpy of Formation of Alkanes." *Indian Journal of Chemistry*, vol. 37A, pp. 849–855, 1998.

[15] B. Zhou, I. Gutman, B. Furtula, and Z. Du, "On two Types of Geometric-Arithmetic Index," *Chemical Physics Letters*, vol. 482, pp. 153–155, 2009.

[16] 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.*, vol. 16, p. 209, 1984.

[17] 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.

[18] N. Rashevsky, "Life, Information Theory, and Topology," *Bull. Math. Biophys.*, vol. 17, pp. 229–235, 1955.

[19] D. Bonchev and N. Trinajstić, "Information Theory, Distance Matrix and Molecular Branching," *J. Chem. Phys.*, vol. 67, pp. 4517–4533, 1977.

[20] S. H. Bertz, "The First General Index of Molecular Complexity," *Journal of the American Chemical Society*, vol. 103, pp. 3241–3243, 1981.

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

[22] A. T. Balaban and T. S. Balaban, "New Vertex Invariants and Topological Indices of Chemical Graphs Based on Information on Distances," *J. Math. Chem.*, vol. 8, pp. 383–397, 1991.

[23] 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.

[24] J. Kim and T. Wilhelm, "What is a Complex Graph?" *Physica A: Statistical Mechanics and its Applications*, vol. 387, no. 11, pp. 2637 – 2652, 2008.

[25] M. Dehmer and L. Sivakumar, "On Distance-Based Entropy Measures," *MATCH Commun. Math. Comput. chem*, vol. MCM12, 2011.

[26] M. Dehmer, "Information Processing in Complex Networks: Graph Entropy and Information Functionals," *Applied Mathematics and Computation*, vol. 201, pp. 82–94, 2008.

[27] M. Dehmer, K. Varmuza, S. Borgert, and F. Emmert-Streib, "On Entropy-based Molecular Descriptors: Statistical Analysis of Real and Synthetic Chemical Structures," *J. Chem. Inf. Model.*, vol. 49, pp. 1655–1663, 2009.

[28] M. Dehmer, F. Emmert-Streib, Y. Tsoy, and K. Varmuza, "Quantifying Structural Complexity of Graphs: Information Measures in Mathematical Chemistry," in *Quantum Frontiers of Atoms and Molecules*, M. Putz, Ed. Nova Publishing, 2010, ch. 18, pp. 479–497.

[29] I. Gutman and B. Zhou, "Laplacian Energy of a Graph," *Linear Algebra and its Applications*, vol. 414, no. 1, pp. 29 – 37, 2006.

[30] E. Estrada, "Characterization of 3D Molecular Structure," *Chemical Physics Letters*, vol. 319, pp. 713–718, 2000.

[31] G. H. Fath-Tabar, A. R. Ashrafi, and I. Gutman, “Note on Estrada and L-Estrada Indices of Graphs,” *Classe des Sciences Mathématiques et Naturelles, Sciences mathématiques naturelles / sciences mathematiques*, vol. CXXXIX, no. 34, pp. 1–16, 2009.

[32] D. Watts, *Small Worlds: The Dynamics of Networks Between Order and Randomness.* Princeton Univ Pr, 2003.

[33] D. Watts and S. Strogatz, “Collective dynamics of ?Small-World? Networks,” *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[34] M. Randic, “On Molecular Identification Numbers,” *Journal of Chemical Information and Computer Sciences*, vol. 24, no. 3, pp. 164–175, 1984.

[35] O. Ivanciuc and A. Balaban, “Design of Topological Indices. Part 3. New Identification Numbers for Chemical Structures: MINID and MINSID,” *Croatica chemica acta*, vol. 69, pp. 9–16, 1996.

[36] M. Randic, “Molecular ID numbers: By Design,” *Journal of Chemical Information and Computer Sciences*, vol. 26, no. 3, pp. 134–136, 1986.

[37] M. Randić and P. Jurs, “On a Fragment Approach to Structure-activity Correlations,” *Quantitative Structure-Activity Relationships*, vol. 8, no. 1, pp. 39–48, 1989.

[38] 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. Elsevier, Amsterdam, 1987, pp. 159–176.

[39] K. Szymanski, W. Müller, J. Knop, and N. Trinajstić, “On the Identification Numbers for Chemical Structures,” *International Journal of Quantum Chemistry*, vol. 30, no. S20, pp. 173–183, 1986.

[40] C. Hu and L. Xu, “On Hall and Kier’s Topological State and Total Topological Index,” *Journal of Chemical Information and Computer Sciences*, vol. 34, no. 6, pp. 1251–1258, 1994.