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

Laurin AJ Mueller, Karl G Kugler, Matthias Dehmer

August 18, 2011

Contents

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 give you an idea 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:

- 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
```

If you have already created networks that you want to analyze with `QuACN`, R offers several ways to import them. (It is important to know that networks have to be represented by *graphNEL*-objects.) Note that there is no general procedure to get your networks into an R-workspace. Some possibilities to import network data are listed below:

- **Adjacency matrix**: A representation of your 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)

[1] "Subgraph distribution:"
cclens.g
 1  3  7
87  2  1

> lcc

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

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.

Note that every descriptor has 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 you want 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. Specially with large networks it saves a lot of time, not to calculate these parameters for each descriptor again, and will enhance the performance of your script.

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

3.1 Descriptors Based on Distances in a Graph

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

Wiener Index [3]:

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

where $|N(G)| := |N|$ denotes the number of Nodes of the complex network. $d(v_i, v_j)$ stands for shortest distances between $v_i, v_j \in V$.

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

[1] 43
```

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] 21.16667
> harary(g, mat.dist)
[1] 21.16667
```

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] 2.414364
> balabanJ(g, mat.dist)
[1] 2.414364
```

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]:

$$\Delta\mu(G) := \frac{1}{N}\sum_{i=1}^{N}|\mu(v_i) - \bar{\mu}|, \tag{4}$$

where

$$\mu(v_i) := \sum_{j=1}^{N} d(v_i, v_j), \tag{5}$$

and

$$\bar{\mu} := \frac{2W}{N}. \tag{6}$$

```
> meanDistanceDeviation(g)
[1] 1.6875
> meanDistanceDeviation(g, mat.dist)
[1] 1.6875
```

Compactness [7]:

$$C(G) := \frac{4W}{|N|(|N| - 1)}. \tag{7}$$

```
> compactness(g)
[1] 3.071429
> compactness(g, mat.dist)
[1] 3.071429
> compactness(g, mat.dist, wiener(g, mat.dist))
[1] 3.071429
```

Product of Row Sums index [8]:

$$\mathrm{PRS}(G) = \prod_{i=1}^{|N|} \mu(v_i) \quad \text{or} \quad \log\big(\mathrm{PRS}(G)\big) = \log\left(\prod_{i=1}^{|N|} \mu(v_i)\right). \tag{8}$$

```
> productOfRowSums(g, log = FALSE)

[1] 157464000

> productOfRowSums(g, log = TRUE)

[1] 27.23045

> productOfRowSums(g, mat.dist, log = FALSE)

[1] 157464000

> productOfRowSums(g, mat.dist, log = TRUE)

[1] 27.23045
```

Hyper-distance-path index [9]

$$D_P(G) := \frac{1}{2}\sum_{i=1}^{|N|}\sum_{j=1}^{|N|} d(v_i, v_j) + \frac{1}{2}\sum_{i=1}^{|N|}\sum_{j=1}^{|N|} \binom{d(v_i, v_j)}{2}. \tag{9}$$

```
> hyperDistancePathIndex(g)

[1] 60

> hyperDistancePathIndex(g, mat.dist)

[1] 60

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

[1] 60
```

3.2 Descriptors Based on Other Graph-Invariants

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

Index of total adjacency [10]:

$$A(G) := \frac{1}{2}\sum_{i=1}^{|N|}\sum_{j=1}^{|N|} a_{ij}. \tag{10}$$

```
> totalAdjacency(g)

[1] 17

> totalAdjacency(g, mat.adj)

[1] 17
```

Zagreb group indices [11]:

$$Z_1(G) := \sum_{i=1}^{|N|} k_{v_i}, \tag{11}$$

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

$$Z_2(G) := \sum_{(v_i,v_j)\in E} k_{v_i} k_{v_j}. \tag{12}$$

```
> zagreb1(g)
[1] 32
> zagreb1(g, vec.degree)
[1] 32
> zagreb2(g)
[1] 298
> zagreb2(g, vec.degree)
[1] 298
```

Randić connectivity index [12]:

$$R(G) := \sum_{(v_i,v_j)\in E} [k_{v_i} k_{v_j}]^{-\frac{1}{2}}. \tag{13}$$

```
> randic(g)
[1] 3.602215
> randic(g, vec.degree)
[1] 3.602215
```

The complexity index B [10]:

$$B(G) := \sum_{i=1}^{|N|} \frac{k_{v_i}}{\mu(v_i)}. \tag{14}$$

```
> complexityIndexB(g)
[1] 3.255556
> complexityIndexB(g, mat.dist)
[1] 3.255556
> complexityIndexB(g, mat.dist, vec.degree)
[1] 3.255556
```

Normalized edge complexity [10]:

$$E_N(G) := \frac{A(G)}{|N|^2}. \tag{15}$$

```
> normalizedEdgeComplexity(g)
[1] 0.265625
> normalizedEdgeComplexity(g, totalAdjacency(g, mat.adj))
[1] 0.265625
```

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 [13, 14]:

$$I^V_{orb}(G) := -\sum_{i=1}^{k} \frac{|N^V_i|}{|N|} \log\left(\frac{|N^V_i|}{|N|}\right). \tag{16}$$

$|N^V_i|$ 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 [15]:

$$I_D(G) := -\frac{1}{|N|}\log\left(\frac{1}{|N|}\right) - \sum_{i=1}^{\rho(G)} \frac{2k_i}{|N|^2}\log\left(\frac{2k_i}{|N|^2}\right), \tag{17}$$

$$I^W_D(G) := W(G)\log(W(G)) - \sum_{i=1}^{\rho(G)} ik_i \log(i). \tag{18}$$

k_i is the occurrence of a distance possessing value i in the distance matrix of G.

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

[1] 1.208931

> bonchev1(g,mat.dist)

[1] 1.208931

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

[1] 170.3098

> bonchev2(g,mat.dist)

[1] 170.3098

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

[1] 170.3098
```

BERTZ complexity index [16]:

$$C(G) := 2N\log(|N|) - \sum_{i=1}^{k} |N_i| \log\left(|N_i|\right). \tag{19}$$

$|N_i|$ are the cardinalities of the vertex orbits as defined in Eqn. (16).

```
> bertz(g)

[1] 42

> bertz(g, mat.dist)

[1] 42

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

[1] 42
```

Radial centric information index [17]:

$$I_{C,R}(G) := \sum_{i=1}^{k} \frac{|N_i^e|}{|N|} \log\left(\frac{|N_i^e|}{|N|}\right). \tag{20}$$

$|N_i^e|$ 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 [17]:

$$I_{deg}(G) := \sum_{i=1}^{\bar{k}} \frac{|N_i^{k_v}|}{|N|} \log\left(\frac{|N_i^{k_v}|}{|N|}\right). \tag{21}$$

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

```
> vertexDegree(g)

[1] 2.25

> vertexDegree(g, vec.degree)

[1] 2.25
```

Balaban-like information indices [18]:

Note, that this class of Descriptors return *Inf* if you have a graph with $|V| < 3$.

$$U(G) := \frac{|E|}{\mu+1} \sum_{(v_i,v_j)\in E} [u(v_i)u(v_j)]^{-\frac{1}{2}}, \tag{22}$$

$$X(G) := \frac{|E|}{\mu+1} \sum_{(v_i,v_j)\in E} [x(v_i)x(v_j)]^{-\frac{1}{2}}, \tag{23}$$

where

$$
\begin{aligned}
u(v_i) &:= -\sum_{j=1}^{\sigma(v_i)} \frac{j|S_j(v_i,G)|}{\mu(v_i)} \log\left(\frac{j}{\mu(v_i)}\right), & (24)\\
x(v_i) &:= -\mu(v_i)\log(d(v_i)) - y_i, & (25)\\
y_i &:= \sum_{j=1}^{\sigma(v_i)} j|S_j(v_i,G)|\log(j), & (26)\\
\mu(v_i) &:= \sum_{j=1}^{|N|} d(v_i,v_j) = \sum_{j=1}^{|N|} j|S_j(v_i,G)|. & (27)
\end{aligned}
$$

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

[1] 8.831362

> balabanlike1(g,mat.dist)

[1] 8.831362

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

[1] 0.8436946

> balabanlike2(g,mat.dist)

[1] 0.8436946
```

Graph vertex complexity index [19]:

$$I_V(G) := \sum_{i=1}^{N} v_i^c, \tag{28}$$

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

$$v_i^c := \sum_{j=0}^{\sigma(v_i)} \frac{k_j^{v_i}}{N} \log\left(\frac{k_j^{v_i}}{N}\right). \tag{29}$$

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

```
> graphVertexComplexity(g)

[1] -12.08022

> graphVertexComplexity(g, mat.dist)

[1] -12.08022
```

3.4 Parametric Graph Entropy Measures

Measures of this group [20, 21] 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. We yield [20],

$$I_f(G) := -\sum_{i=1}^{|N|} \frac{f(v_i)}{\sum_{j=1}^{|N|} f(v_j)} \log\left(\frac{f(v_i)}{\sum_{j=1}^{|N|} f(v_j)}\right), \tag{30}$$

where $I_f(G)$ represents a family of graph entropy [20] measures depending on the information functional. Further we implemented the following measurement[21]:

$$I_f^{\lambda}(G) := \lambda \left(\log(|N|) + \sum_{i=1}^{|N|} p(v_i) \log(p(v_i)) \right), \tag{31}$$

$$p(v_i) := \frac{f(v_i)}{\sum_{j=1}^{|N|} f(v_j)}, \tag{32}$$

where $p^V(v_i)$ are the vertex probabilities, $\lambda > 0$ a scaling constant. This measure can be interpreted as the distance between the entropy defined in equation 30 and maximum entropy ($\log(|N|)$).

We integrated 4 different information functionals:

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

$$f^V(v_i) := c_1|S_1(v_i,G)| + c_2|S_2(v_i,G)| + \cdots + c_{\rho(G)}|S_{\rho(G)}(v_i,G)|, \tag{33}$$

   where $c_k > 0$.

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

$$f^P(v_i) := c_1 l(P(L_G(v_i,1))) + c_2 l(P(L_G(v_i,2))) + \cdots + c_{\rho(G)} l(P(L_G(v_i,\rho(G)))), \tag{34}$$

   where $c_k > 0$.

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

$$f^C(v_i) := c_1 \beta^{L_G(v_i,1)}(v_i) + c_2 \beta^{L_G(v_i,2)}(v_i) + \cdots + c_{\rho(G)} \beta^{L_G(v_i,\rho(G))}(v_i), \tag{35}$$

   where $c_k > 0$.

4. Calculates the degree degree association index(*"degree"*) [22]:

$$f^{\Delta}(v_i) := \alpha^{c_1 \Delta^G(v_i,1) + c_2 \Delta^G(v_i,2) + \cdots + c_{\rho(G)} \Delta^G(v_i,\rho(G))}, \tag{36}$$

   where $c_k > 0$, $1 \leq k \leq \rho(G)$ and $\alpha > 0$

We implemented 4 different settings (as example settings) of the weighting parameter c_i:

1. constant

$$c_1 := 1,\ c_2 := 1, \cdots,\ c_{\rho(G)} := 1. \tag{37}$$

2. linear

$$c_1 := \rho(G),\ c_2 := \rho(G) - 1, \cdots,\ c_{\rho(G)} := 1. \tag{38}$$

3. quadratic

$$c_1 := \rho(G)^2, c_2 := (\rho(G) - 1)^2, \cdots, c_{\rho(G)} := 1. \tag{39}$$

4. exponential

$$c_1 := \rho(G), c_2 := \rho(G)e^1, \cdots, c_{\rho(G)} := \rho(G)e^{-\rho(G)+1}. \tag{40}$$

   $\rho(G)$ represents the diameter of the network.

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" you have to specify your customized weighting schema with the parameter *custCoeff*.

- *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. To use it you need to set *coeff*="const".

Note that some combinations of these settings can cause the descriptor to return *NaN*. In that case you have to check for *warnings*.

The method returns a list with following entries:

- *entropy*: contains the entropy, see formula 30
- *distance*: contains the distance described in formula 31
- *pis*: contains the probability distribution, see formula 32
- *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] 2.990011

$distance
[1] 9.9892

$pis
        1         2         3         4         5         6         7         8
0.1376812 0.1304348 0.1304348 0.1376812 0.1376812 0.0942029 0.1159420 0.1159420

$fvis
 1  2  3  4  5  6  7  8
19 18 18 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
```

3.5 Eigenvalue-based Descriptors

This class contains Eigenvalue-based Descriptors proposed in Dehmer et. al [23].

$$H_{M_s(G)} = \sum_{i=1}^{k} \frac{|\lambda_i|^{\frac{1}{s}}}{\sum_{j=1}^{k} |\lambda_j|^{\frac{1}{s}}} \log \left(\frac{|\lambda_i|^{\frac{1}{s}}}{\sum_{j=1}^{k} |\lambda_j|^{\frac{1}{s}}} \right), \tag{41}$$

$$S_{M_s(G)} = |\lambda_1|^{\frac{1}{s}} + |\lambda_2|^{\frac{1}{s}} + \ldots + |\lambda_k|^{\frac{1}{s}}, \tag{42}$$

$$IS_{M_s(G)} = \frac{1}{|\lambda_1|^{\frac{1}{s}} + |\lambda_2|^{\frac{1}{s}} + \ldots + |\lambda_k|^{\frac{1}{s}}}, \tag{43}$$

$$P_{M_s(G)} = |\lambda_1|^{\frac{1}{s}} \cdot |\lambda_2|^{\frac{1}{s}} \ldots |\lambda_k|^{\frac{1}{s}}, \tag{44}$$

$$IP_{M_s(G)} = \frac{1}{|\lambda_1|^{\frac{1}{s}} \cdot |\lambda_2|^{\frac{1}{s}} \ldots |\lambda_k|^{\frac{1}{s}}}, \tag{45}$$

With 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] 2.91436

$SMs
[1] 9.912979

$ISMs
[1] 0.1008779

$PMs
[1] 3.464102

$IPMs
[1] 0.2886751
```

2. Laplacian matrix

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

$HMs
[1] 2.731375

$SMs
[1] 14.73375

$ISMs
[1] 0.06787137

$PMs
[1] 1.465232e-06

$IPMs
[1] 682485.6
```

3. Distance matrix

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

```
$HMs
[1] 2.800874

$SMs
[1] 11.81693

$ISMs
[1] 0.08462436

$PMs
[1] 7.745967

$IPMs
[1] 0.1290994
```

4. Distance path Matrix

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

$HMs
[1] 2.776074

$SMs
[1] 14.61724

$ISMs
[1] 0.06841237

$PMs
[1] 36.29049

$IPMs
[1] 0.02755542
```

5. Augmented vertex degree matrix

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

$HMs
[1] 2.805871

$SMs
[1] 13.9841

$ISMs
[1] 0.07150979

$PMs
[1] 31.11596

$IPMs
[1] 0.03213785
```

6. Extended adjacency matrix

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

$HMs
[1] 2.926072
```

```
$SMs
[1] 10.9429

$ISMs
[1] 0.09138349

$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] 0.8482934

$SMs
[1] 3.277543
```

```
$ISMs
[1] 0.3051066

$PMs
[1] 4.932483e-31

$IPMs
[1] 2.027376e+30
```

10. Weighted structure function matrix IM_2

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

    $HMs
    [1] 1.02449

    $SMs
    [1] 3.432103

    $ISMs
    [1] 0.2913665

    $PMs
    [1] 4.507051e-35

    $IPMs
    [1] 2.218746e+34
    ```

For a detailed description of this class see Dehmer et. al [23].

4 Session Info

```
> sessionInfo()

R version 2.13.1 Patched (2011-08-17 r56750)
Platform: x86_64-unknown-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_1.3.3    Rmpfr_0.4-2    combinat_0.0-8 igraph_0.5.5-2 RBGL_1.28.0
[6] graph_1.30.0

loaded via a namespace (and not attached):
[1] tools_2.13.1
```

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. NovÃÍre, 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. Jäntschi, *Molecular Topology.* Nova Publishing, 2001, new York, NY, USA.

[12] 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.

[13] 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.

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

[15] D. Bonchev and N. Trinajstić, "Information theory, distance matrix and molecular branching," *J. Chem. Phys.*, vol. 67, pp. 4517–4533, 1977.

[16] S. H. Bertz, "The first general index of molecular complexity," *Journal of the American Chemical Society*, vol. 103, pp. 3241–3243, 1981.

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

[18] 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.

[19] 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.

[20] M. Dehmer, "Information processing in complex networks: Graph entropy and information functionals," *Applied Mathematics and Computation*, vol. 201, pp. 82–94, 2008.

[21] 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.

[22] 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, to appear.

[23] M. Dehmer, F. Emmert-Streib, Y. R. Tsoy, and K. Varmuza, "Quantifying structural complexity of graphs: Information measures in mathematical chemistry," pp. 467–485, 2010.