Using QuACN to analyze complex biological networks

Laurin Müller

August 3, 2010

Contents

1 Overview

This vignette will give an overview about the usage of QuACN.

Chapter **??** will give you an idea how to import already exiting networks. In Chapter **??** a brief description of the implemented measures is given, and it shows how to call the related method in R.

2 Networks

```
> library("QuACN")
> set.seed(666)
> g <- randomGraph(1:8, 1:5, 0.36)
> plot(g, "neato")
> g
```

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

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

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. Find some possibilities to import network data listed below:

- **Adjacency matrix**: An 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.
- **Graph Markup Language(GML)**: The *graph*-package offers the possibility to import graphs that a represented in GML.
- **System Biology Markup Language(SBML)**: With *RSBML*-package it is possible to import SBML-Models.
- **igrah-package**: Networks created with the *igraph*-package can be converted into graphNEL objects.

Figure 1: A Random graph to show the functionality of the methods.

3 Network Descriptors

This section will give 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 paramater stays emty 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 recomended 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 can save 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

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

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

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

```
> harary(g)
```

```
[1] 21.16667
```

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

```
[1] 21.16667
```

Balaban J Index

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

```
> 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.

```
> meanDistanceDeviation(g)

[1] 1.6875

> meanDistanceDeviation(g, mat.dist)

[1] 1.6875
```

Compactness

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

```
> 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

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

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

[1] 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

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

```
> hyperDistancePathIndex(g)

[1] 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

$$A(G) := \frac{1}{2}\sum_{i=1}^{|N|}\sum_{j=1}^{|N|} a_{ij}, \tag{7}$$

```
> totalAdjacency(g)
[1] 17
> totalAdjacency(g, mat.adj)
[1] 17
```

Zagreb group indices

$$Z_1(G) := \sum_{i=1}^{|N|} k_{v_i}, \tag{8}$$

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

$$Z_2(G) := \sum_{(v_i,v_j)\in E} k_{v_i} k_{v_j} \tag{9}$$

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

Randić index

$$R(G) := \sum_{(v_i,v_j)\in E} [k_{v_i} k_{v_j}]^{-\frac{1}{2}} \tag{10}$$

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

The complexity index B

$$B(G) := \sum_{i=1}^{|N|} \frac{k_{v_i}}{\mu(v_i)}. \tag{11}$$

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

Normalized edge complexity

$$E_N(G) := \frac{A(G)}{|N|^2} \tag{12}$$

```
> 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

$$I_{orb}^V(G) := -\sum_{i=1}^{k} \frac{|N_i^V|}{|N|} \log\left(\frac{|N_i^V|}{|N|}\right). \tag{13}$$

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

```
> topologicalInfoContent(g)
[1] 2.25
> topologicalInfoContent(g, mat.dist)
[1] 2.25
> topologicalInfoContent(g, mat.dist, vec.degree)
[1] 2.25
```

Bonchev - Trinajstić indices

$$I_D(G) := -\frac{1}{|N|} \log\left(\frac{1}{|N|}\right) - \sum_{i=1}^{\rho(G)} \frac{2k_i}{|N|^2} \log\left(\frac{2k_i}{|N|^2}\right), \tag{14}$$

$$I_D^W(G) := W(G)\log(W(G)) - \sum_{i=1}^{\rho(G)} ik_i \log(i). \tag{15}$$

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

$$C(G) := 2N \log(|N|) - \sum_{i=1}^{k} |N_i| \log(|N_i|). \tag{16}$$

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

```
> bertz(g)
[1] 42
> bertz(g, mat.dist)
[1] 42
> bertz(g, mat.dist, vec.degree)
[1] 42
```

Radial centric information index

$$I_{C,R}(G) := \sum_{i=1}^{k} \frac{|N_i^e|}{|N|} \log\left(\frac{|N_i^e|}{|N|}\right). \tag{17}$$

$|N_i^e|$ 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

$$I_{deg}(G) := \sum_{i=1}^{\bar{k}} \frac{|N_i^{k_v}|}{|N|} \log\left(\frac{|N_i^{k_v}|}{|N|}\right). \tag{18}$$

$|N_i^{k_v}|$ 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

$$U(G) := \frac{|E|}{\mu + 1} \sum_{(v_i, v_j) \in E} [u(v_i)u(v_j)]^{-\frac{1}{2}}, \tag{19}$$

$$X(G) := \frac{|E|}{\mu + 1} \sum_{(v_i, v_j) \in E} [x(v_i)x(v_j)]^{-\frac{1}{2}}, \tag{20}$$

where

$$u(v_i) := -\sum_{j=1}^{\sigma(v_i)} \frac{j|S_j(v_i, G)|}{\mu(v_i)} \log\left(\frac{j}{\mu(v_i)}\right), \tag{21}$$

$$x(v_i) := -\mu(v_i) \log(d(v_i)) - y_i, \tag{22}$$

$$y_i := \sum_{j=1}^{\sigma(v_i)} j|S_j(v_i, G)| \log(j), \tag{23}$$

$$\mu(v_i) := \sum_{j=1}^{|N|} d(v_i, v_j) = \sum_{j=1}^{|N|} j|S_j(v_i, G)|. \tag{24}$$

```
> #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
```

Vertex degree equality-based information indes

$$\bar{I}_{\deg}(G) := \sum_{i=1}^{\bar{k}} \frac{|N_i^{k_v}|}{N} \log\left(\frac{|N_i^{k_v}|}{N}\right). \tag{25}$$

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

```
> graphVertexComplexity(g)

[1] -12.08022

> graphVertexComplexity(g, mat.dist)

[1] -12.08022
```

3.4 Parametric Graph Entropy Measures

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. We yield [**?**],

$$I_f(G) := -\sum_{i=1}^{|N|} \frac{f(v_i)}{\sum_{j=1}^{|N|} f(v_j)} \log\left(\frac{f(v_i)}{\sum_{j=1}^{|N|} f(v_j)}\right), \tag{26}$$

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

$$I_f^{\lambda}(G) := \lambda\left(\log(|N|) + \sum_{i=1}^{|N|} p(v_i)\log(p(v_i))\right), \tag{27}$$

$$p(v_i) := \frac{f(v_i)}{\sum_{j=1}^{|N|} f(v_j)}, \tag{28}$$

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 **??** and maximum entropy ($\log(|N|)$).

We integrated 3 different information functionals:

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

$$f(v_i) := c_1|S_1(v_i, G)| + c_2|S_2(v_i, G)| + \cdots + c_{\rho(G)}|S_{\rho(G)}(v_i, G)|, \tag{29}$$

   where $c_k > 0$.

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

$$f^{P_2}(v_i) := c_1 l(P(L_G(v_i,1))) + c_2 l(P(L_G(v_i,2))) + \cdots + c_{\rho(G)} l(P(L_G(v_i,\rho(G)))), \tag{30}$$

where $c_k > 0$.

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

$$f^{C_2}(v_i) := c_1 \beta^{L_G(v_i,1)}(v_i) + c_2 \beta^{L_G(v_i,2)}(v_i) + \cdots + c_{\rho(G)} \beta^{L_G(v_i,\rho(G))}(v_i), \tag{31}$$

where $c_k > 0$.

We implemented 4 different settings of the weighting parameter c_i:

1. constant
$$c_1 := 1,\, c_2 := 1, \ldots,\, c_{\rho(G)} := 1, \tag{32}$$

2. linear
$$c_1 := \rho(G),\, c_2 := \rho(G) - 1, \ldots,\, c_{\rho(G)} := 1, \tag{33}$$

3. quadratic
$$c_1 := \rho(G)^2, c_2 := (\rho(G) - 1)^2, \ldots, c_{\rho(G)} := 1. \tag{34}$$

4. exponential
$$c_1 := \rho(G), c_2 := \rho(G)e^{-1}, \ldots, c_{\rho(G)} := \rho(G)e^{-\rho(G)+1}. \tag{35}$$

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

To call this type of metwork measure we prvide the method *infoTheoreticGCM*. It has follwing 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" or "localprop" are availiable settings.
- *lamda*: scaling constant for the distance, default set to 1000.
- *custCoeff*: specifies the customized weighting schema. To use it you need to set *coeff*="const".

The method returns a list with following entries:

- *entropy*: contains the entropy, see formula **??**
- *distance*: contains the distance described in formula **??**
- *pis*: contains the probability distribution, see formula **??**
- *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",
+     lamda = 1000)
> l3 <- infoTheoreticGCM(g, mat.dist, coeff = "exp", infofunct = "sphere",
+     lamda = 1000)
> l4 <- infoTheoreticGCM(g, mat.dist, coeff = "const", infofunct = "pathlength",
+     lamda = 4000)
> l5 <- infoTheoreticGCM(g, mat.dist, coeff = "quad", infofunct = "localprop",
+     lamda = 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 
```

> *15*

```
$entropy
[1] 2.924897

$distance
[1] 75.10322

$pis
         1          2          3          4          5          6          7 
0.16666667 0.12851406 0.12851406 0.16666667 0.15160643 0.04518072 0.10642570 
         8 
0.10642570 

$fvis
       1        2        3        4        5        6        7        8 
55.33333 42.66667 42.66667 55.33333 50.33333 15.00000 35.33333 35.33333 
```