# Using the different distances

*Claire Donnat*

*1/19/2018*

This notebook just provides a quick overview of the different distances that have been compared.

## Graph generation

Graphs can be generated by calling the function **generate_realistic_adjacency**, which can generate 4 types of graph (through the igraph library) by changing the **opts** parameter:

- **opts=0: ER-random graph**, with parameters $N$ (number of nodes), and $p$ (probability of connection).
- **opts=1: Barabasi and Albert-random graph** with parameters $N$ (number of nodes), and *power*.
- **opts=2: Island graph**: clique model in which every island is an ER graph, and islands are connected through a pre-specified number of edges. The parameters for this model are *islands.n* (number of islands), *islands.size* (list providing the number of nodes per island), *islands.pin*: list providing the intra-island connectivity (probability of an edge between two nodes in the same island), and *n.inter*: number of edges between each island and the rest (see igraph documentation for additional details)
- 
- **opts=3: Dot-Product-random graph** with parameters $K$ (the dimension of the hidden embedding vector: see igraph documentation for additional details)
- **opts=4: SBM-random graph** with parameters *block.sizes* (list providing the size of each block) and pm (connection probability matrix). This is in spirit similar to the island graph, but provides in general denser graphs.

```
## [1] "Working directory set to /Users/cdonnat/Dropbox/TrackingNetworkChanges"
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
##
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
## Warning: package 'Matrix' was built under R version 3.3.2
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     ginv
```

```
## Loading required package: statnet.common
```

```
## Loading required package: network
```

```
## network: Classes for Relational Data
## Version 1.13.0 created on 2015-08-31.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##                     Mark S. Handcock, University of California -- Los Angeles
##                     David R. Hunter, Penn State University
##                     Martina Morris, University of Washington
##                     Skye Bender-deMoll, University of Washington
##  For citation information, type citation("network").
##  Type help("network-package") to get started.
```

```
##
## Attaching package: 'network'
```

```
## The following objects are masked from 'package:igraph':
##
##     %c%, %s%, add.edges, add.vertices, delete.edges,
##     delete.vertices, get.edge.attribute, get.edges,
##     get.vertex.attribute, is.bipartite, is.directed,
##     list.edge.attributes, list.vertex.attributes,
##     set.edge.attribute, set.vertex.attribute
```

```
## sna: Tools for Social Network Analysis
## Version 2.4 created on 2016-07-23.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##  For citation information, type citation("sna").
##  Type help(package="sna") to get started.
```

```
##
## Attaching package: 'sna'
```

```
## The following objects are masked from 'package:igraph':
##
##     betweenness, bonpow, closeness, components, degree,
##     dyad.census, evcent, hierarchy, is.connected, neighborhood,
##     triad.census
```

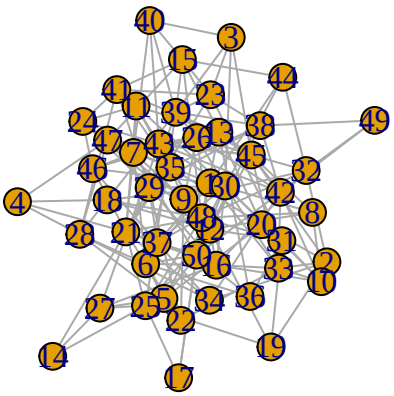
```
## [1] "All functions loaded."
```

Here are a few examples:

- ER graph (opts=0):

```
N=50
Adj=generate_realistic_adjacency(N,opts=0,args=list(),verbose=TRUE,p=0.16)
```

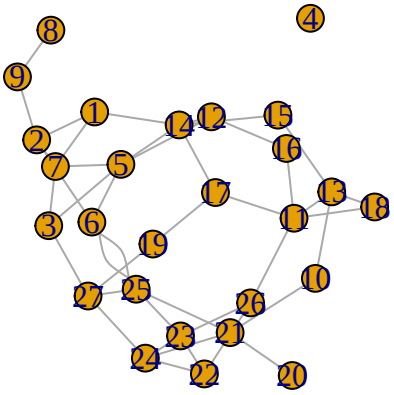
```
## [1] "Type of graph generated:  ER"
```



+PA graph

```
G=generate_realistic_adjacency(N,opts=2,args=list(),verbose=TRUE,power=2.4)
```

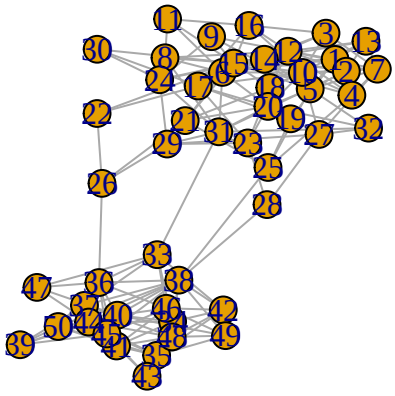
```
## [1] "Type of graph generated:  Island"
## [1] "island graph: islands.n= 3 9"
```



+SBM graph

```
G=generate_realistic_adjacency(N,opts=4,args=list(),verbose=TRUE,pm=cbind( c(0.4,0.1, 0.001), c(.1,0.2,
```

```
## [1] "Type of graph generated:  SBM"
## [1] 50
## [1] "stochastic block model: block size 10"
## [2] "stochastic block model: block size 10"
## [3] "stochastic block model: block size 10"
```



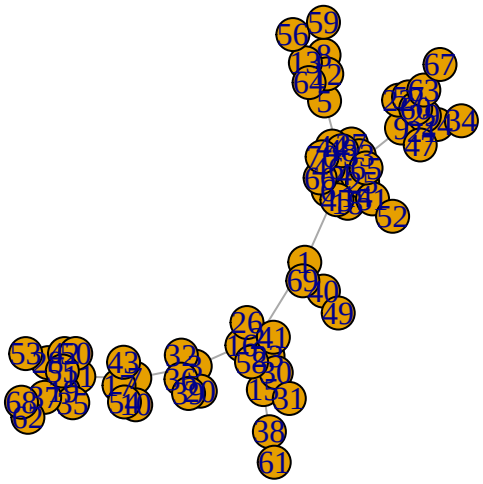
## Using the different distances

Loading all the functions provides a wrapper which (hopefully), makes it easier to select and compute distances. Options include:

- the Hamming, IM and Mikhailov distances, computed by neettools.

```
#### Generate first matrix
N=70
args_l<-list(p=0.1,power=1.7,islands.n=3,islands.size=9,islands.pin=0.3,n.inter=3,K=6,block.sizes=c(10,
Ag<-generate_realistic_adjacency(N,opts=1, args_l=args_l,verbose=TRUE)
```

```
## [1] "Type of graph generated:  Power Law"
## [1] "power graph: p= 1.7"
```



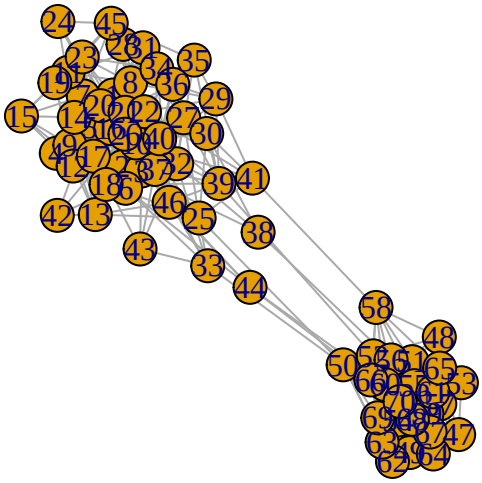
```
A<-as(get.adjacency(Ag),"matrix")

#### Generate evolution
prop=0.3
p=0.4
A_new<-random_alteration_adjacency(A,prop,p)
dist=netdist(A, A_new,d = "HIM")
print(dist)
```

```
##           H          IM         HIM
## 0.005797101 0.027627393 0.019960953
```

- Spanning Tree distances

```
#### Generate first matrix
N=70
args_l<-list(p=0.1,power=1.7,islands.n=3,islands.size=9,islands.pin=0.3,n.inter=3,K=6,block.sizes=c(10,
Ag<-generate_realistic_adjacency(N,opts=4, args_l=args_l,verbose=TRUE)
```

```
## [1] "Type of graph generated:  SBM"
## [1] 70
## [1] "stochastic block model: block size 10"
## [2] "stochastic block model: block size 10"
## [3] "stochastic block model: block size 10"
```



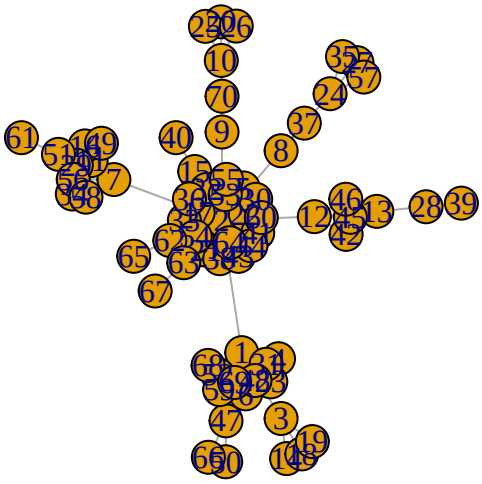
```
A<-as(get.adjacency(Ag),"matrix")

#### Generate evolution
prop=0.3
p=0.4
A_new<-random_alteration_adjacency(A,prop,p)
sp_Anew=get_number_spanning_trees(A_new)
sp_A=get_number_spanning_trees(A)
#print(c(sp_A,sp_Anew))
dist=abs(log(max(sp_A,1))-log(max(sp_Anew,1)))
print(dist)
```

```
## [1] 0.009812703
```

- Polynomial Distances, which require the specification of the downweighting factor $\alpha$ and the order of the polynomial:

```
#### Generate first matrix
N=70
args_l<-list(p=0.1,power=1.7,islands.n=3,islands.size=9,islands.pin=0.3,n.inter=3,K=6,block.sizes=c(10,
Ag<-generate_realistic_adjacency(N,opts=1, args_l=args_l,verbose=TRUE)
```

```
## [1] "Type of graph generated:  Power Law"
## [1] "power graph: p= 1.7"
```



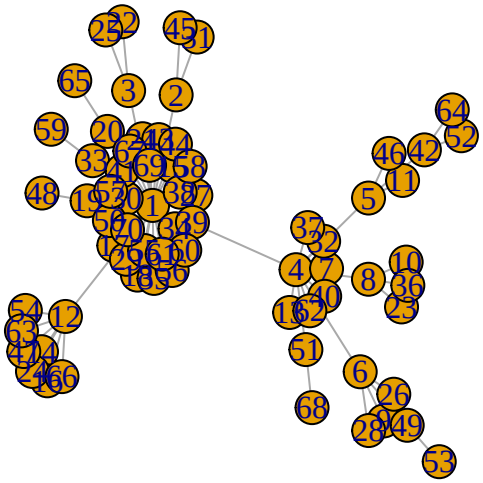
```
A<-as(get.adjacency(Ag),"matrix")

#### Generate evolution
prop=0.3
p=0.4
dist=poly_distance(A, A_new,order_max=3,alpha=0.9)
print(dist)
```

```
## [1] 0.2250028
```

- Eigenvalue-based distances, which require the specification of the filter:

```
#### Generate first matrix
N=70
args_l<-list(p=0.1,power=1.7,islands.n=3,islands.size=9,islands.pin=0.3,n.inter=3,K=6,block.sizes=c(10,
Ag<-generate_realistic_adjacency(N,opts=1, args_l=args_l,verbose=TRUE)
```

```
## [1] "Type of graph generated:  Power Law"
## [1] "power graph: p= 1.7"
```



```
A<-as(get.adjacency(Ag),"matrix")

#### Generate evolution
prop=0.3
p=0.4
dist=eigen_distance(A, A_new,function(x){ifelse(x<2,x,0)})
print(dist)
```

```
## [1] 8.322899
```