TP 1 : Igraph/R - Introduction

Rushed Kanawati

December 2, 2019

Abstract

The goal of this training is to master the *igraph* API: a graph analysis package.

igraph : installation

`igraph` is an open source graph analysis and visualisation package (see `http://igraph.sourceforge.net/`). The package is available for both `Python` and `R` environments. In the curent training we'll use it in a `R` environment.

In your `R` studio shell execute te following instructions :

1. `install.packages("igraph")` # this installs the package in your environment
2. `library(igraph)` # to load the library in the execution environment
3. `help(igraph)` # this provides basic informations about the package

Graph generation & loading

Graph construction

The function `graph.empty()` generates an empty graph :

```
g <- graph.empty(directed=FALSE)
```

you can add nodes and links using the following functions : `add.vertices` and `add.edges`.

```
g <- add.vertices(g,4)
```

```
g <- add.edges(g,c(1,2))
```

You can delete nodes and links using `delete.vertices` et `delete.edges`.

Function `summary(g)` returns a textual description of graph g `plot(g)` allow to plot the graph.

Simple graph generation

`igraph` provides a bunch of graph generation functions. Some exemples are given here:

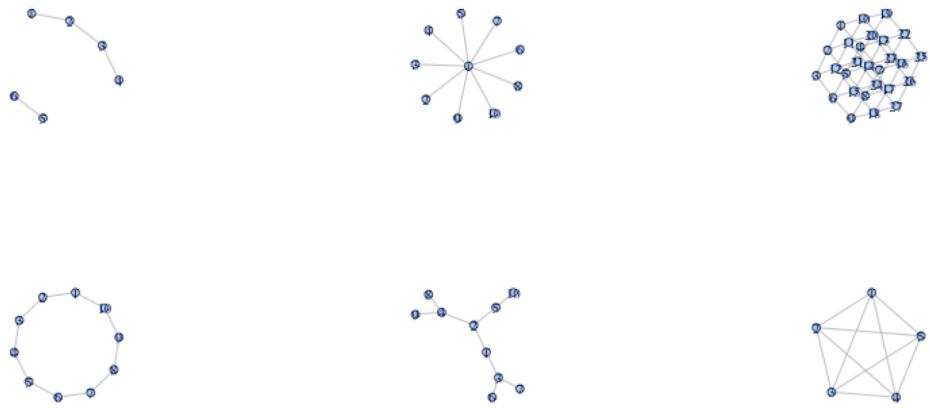

Figure 1: Exemples of graph generation

```
graph( c(1,2,2,3,3,4,5,6), directed=FALSE )
graph.star(10, mode="undirected")
graph.lattice(c(3,3,3), directed=FALSE)
graph.ring(10, directed=FALSE)
graph.tree(10, 2, mode="undirected")
graph.full(5, loops=FALSE, directed=FALSE)
```

Figure 1 illustrates the graphs generated by the previous commands.

Random graphs

`igraph` offers also a set random graphs generation models: *Erdös-Renyi* model, small-world graph model (*Watts-Strogratz*) and preferential attachement (*Barabasi-Albert*). Some exemples are given below:

```
er_graph <- erdos.renyi.game(100, 2/100)
ws_graph <- watts.strogatz.game(1, 100, 4, 0.05)
ba_graph <- barabasi.game(100)
```

Reading graphs

The function `read.graph` allows to read a graph saved in a file. The graph format is specified by the parameter *format*. Exemples:

```
karate <- read.graph("http://cneurocvs.rmki.kfki.hu/igraph/karate.net", format="pajek")
setwd( working directory)
football <- read.graph("football.gml", format="gml")
```

The function `write.graph` saves a graph in one of the supported format.

Graph metrics

`igraph` provides a rich set of metric functions. Main functions are:

- `vcount(g)` : number of nodes in g
- `ecount(g)` : number of links in g
- `graph.density(g)` : returns the density of g
- `diameter(g)` : returns the diameter of g
- `degree(g)` : returns the degree list of nodes in g
- `degree.distribution(g)` : computes the degree distribution of g
- `transitivity(g)` : returns the clustering-coefficient of g
- `shortest.paths(g)` : returns a matrix where elements i,j are the length of shortest paths between nodes i and j.
- `betweenness(g)` : computes the betweenness centrality.
- `closeness(g)` : computes the closeness centrality.
- `is.connected(g)` : returns `TRUE` if g is connected.
- `clusters(g)` : returns the list of connected components in g
- `neighbors(g,x)` : returns the list of neighbors of node x in graph g

Graph visualisation

Function *plot*(g) plots a graph. This function admets a large set of parameters in order to customize the plot. `V(g)` (resp. `E(g)`) is the list of nodes (resp. links) in `g`. we can add to a node (resp. a link) a set of attributes. For exemple, we can add the attribute `color` as follows:

```
V(g)$color <- sample( c("red", "black"), vcount(g), rep=TRUE)
E(g)$color <- "grey"
plot(g)
```

Attribut `shape` can be used to set the shape of a node (ex. circle, square). Attribut `layout` determines the visualisation algorithm to applied for graph plotting. Exemple :

```
g <- graph.ring(10)
g$layout <- layout.circle
plot(g)
```

Exercices

1. Develop a R script that returns the basic topological features of a complex network.
2. Develop a script that plot the degree distribution of a graph.
3. Study the variation of graph diameter and transitivity for different random graph models in function of the number of nodes. Conclusion ?
4. Download the following files `dolphins.gml`, `polblogs.gml`, `football.gml`, `karate.gml` from `http://lipn.fr/~kanawati/ars`. In these graphs the ground-truth community of each node is given by the attribut `value`. Plot these graphs by coloring each community with a different color. (NB. use the function `rainbow` for generating different colors)
5. Plot these graphs by changing the size of nodes in function of degree centrality and then in function of the closeness centrality.
6. From each graph, extract a sub-graph centered on the most central node. Apply different centrality measures.