Analyse des réseaux sociaux avec iGraph & R

Ahmed Mohameden
amed.mohameden@gmail.com

Université Cheikh Anta Diop
Faculté des Sciences et Techniques
Département de Mathématiques et Informatique
Section Informatique

9 Avril 2018

Objectifs du cours

A la fin de ce cours, vous serez en mesure de :

1. Créer, générer ou modifier un graphe ;
2. Calculer et interpréter les mesures locales et globales ;
3. Visualiser les graphes selon plusieurs points de vue ;
4. Distinguer les formats des fichiers supportés par iGraph ;
5. Importer/exporter des données.

Plan

Création des graphes

La création d'un graphe peut se faire de deux manières :

1. Création manuelle : c'est-à-dire qu'on procède à la construction du graphe en ajoutant, tout d'abord, les nœuds. Ensuite, on ajoute les liens ;
2. Génération des graphes, il s'agit d'appeler les fonctions prédéfinies de iGraph pour construire le graphe. Le graphe généré peut être déterministe ou non.

Création manuelle

La création manuelle d'un graphe s'effectue en 3 étapes :

1. Création d'un graphe vide

```
g <- make_empty_graph()
```

2. Ajout des sommets au graphe

```
g <- add_vertices(g, 1, name = "Idy", color = "green")
g <- add_vertices(g, 4,
     name = c("Ahmed", "Khaled", "Danish", "Cazabet"),
     color = c("orange", "blue", "yellow", "black"))
```

3. Ajoute des liens

```
g <- add_edges(g, c(1, 2, 1, 4, 1, 1, 5, 2, 3, 2,
                    1, 2, 6, 3, 5, 4, 1, 6))
```

Génération des graphes

La génération des graphes peut être faite de deux manières :

1. Graphe déterministe : un graphe est dit déterministe s'il garde la même structure à chaque génération. La structure des graphes déterministes est connue à l'avance. Exemples : graphe complet, graphe cerclé, graphe étoilé, etc. ;
2. Graphe non déterministe : la structure de ce type de graphes change à chaque génération. Autrement, l'existence d'un lien dépend d'une probabilité $p \in]0, 1[$.

Graphes déterministes

iGraph permet de générer plusieurs types des graphes déterministes dont nous citons :

- Graphe cerclé

```
g <- make_ring(n=10, directed = FALSE, mutual = FALSE)
```

- Graphe étoilé

```
g <- make_star(n = 10, mode = "out")
```

- Graphe complet

```
g <- make_full_graph(n = 10, directed = TRUE, loops = FALSE)
```

Graphes non déterministes (1/2)

Dans un graphe non déterministe, l'existence d'un lien est le résultat d'un processus probabiliste. Les graphes non déterministes permettent de compléter un réseau tout en respectant ses caractéristiques globales (distribution des degrés, distance moyenne, etc.). Ils sont utilisés également dans l'étude des différentes compositions possibles des réseaux.

Graphes non déterministes (2/2)

Dans ce cours, nous nous intéressons à deux types de graphes non déterministes :

1. Graphes petit-monde
   - Distance moyenne très courte (six degrés de séparation) ;
   - Coefficient de regroupement élevé (transitivité des liens) ;
   - Structure en communautés (les nœuds partagent des propriétés similaires ou des intérêts communs tendent à être plus connectés entre eux qu'avec le reste du réseau) ;
   - Méthode de génération : Modèle Watts-Strogatz ;
2. Graphes sans échelle
   - Principe de l'attachement préférentiel (les riches deviennent de plus en plus riches) ;
   - Existence de hubs (nœuds ego-centrés, ponts de passage, etc.) ;
   - Méthode de génération : Modèle Barabasi-Albert.

Comportement des graphes

Le tableau ci-dessous présente une comparaison des graphes suivant leur comportement :

	Structure	Caractéristiques
Graphe déterministe	fixe	fixes
Graphe non déterministe	variante	fixes
Graphe aléatoire	variante	variantes

Nous parlons de « graphes aléatoires » si nous n'imposons pas des contraintes ni sur la structure, ni sur les caractéristiques du graphe. Le modèle Erdös-Rényi permet de générer des graphes de ce type.

Informations sur le graphe (1/3)

La fonction `summary` permet de décrire un objet iGraph. Prenons l'exemple suivant :

```
IGRAPH DNWB 14 20 -- Ring graph
+ attr: loops (g/l), id (v/n), color (e/c)
```

Comme illustré par cet exemple, la description commence par le mot IGRAPH indiquant qu'il s'agit d'un objet de la classe iGraph. Ensuite, nous constatons la présence de 4 lettres :

- **D** ou **U**, qui désigne que le graphe est orienté ou non orienté ;
- **N** qui indique que le graphe est nommé, c'est à dire que les nœuds possèdent un attribut « name » ;
- **W** qui signifie que le graphe est pondéré (les liens possèdent un attribut nommé « weight ») ;
- **B** qui montre que le graphe est bipartite.

Informations sur le graphe (2/3)

```
 IGRAPH DNWB 14 20 -- Ring graph
+ attr: loops (g/l), id (v/n), color (e/c)
```

Les deux nombres qui suivent désignent, respectivement, le nombre de nœuds et le nombre de liens. Le reste de la description présente les attributs du graphe (g), les attributs des nœuds (v) et les attributs des liens (e). Le type d'attribut peut être une valeur booléenne (l), un nombre (n) ou une chaine de caractères (c).

Informations sur le graphe (3/3)

Pour recueillir des informations sur la structure du graphe, utilisez les fonctions suivantes :

- Nœuds
  - liste/nombre des nœuds : `V(g)`, `vcount(g)`
  - noms/valeurs des attributs des nœuds : `vertex_attr_names(g)`, `vertex_attr(g)`
- Liens
  - liste/nombre des liens : `E(g)`, `ecount(g)`
  - noms/valeurs des attributs des liens : `edge_attr_names(g)`, `edge_attr(g)`
- Le graphe est-il simple ou non ? `is.simple(g)`
- Matrice d'adjacence : `g[]`

Modification des graphes

La modification du graphe peut se faire de deux manières :

1. Modification structurelle ;
2. Modification visuelle.

Modification structurelle

La modification structurelle d'un graphe désigne le changement de sa structure. On distingue 4 types de modifications structurelles :

- Ajout/suppression de nœuds ;
- Ajout/suppression de liens ;
- Modification de la direction de liens ;
- Modification des poids de liens.

Modification visuelle

La modification visuelle désigne le changement des valeurs des attributs relatifs à la représentation du graphe. Ci-dessous quelques exemples :

- Modification des valeurs des attributs
  - Rectification des noms de quelques nœuds ;
  - Agrandir la taille des liens suivant la valeur du poids ;
- Ajout/suppression des attributs
  - Ajout d'un attribut « nom » pour les nœuds ;
  - Ajout d'un attribut « couleur » pour les liens ;
  - Suppression de l'attribut « taille » pour les nœuds.

Visualisation des graphes

iGraph offre 3 types de visualisation des graphes :

1. Visualisation non interactive

```
plot(g, vertex.color = "green", edge.color = "blue")
```

2. Visualisation interactive

```
tkplot(g)
```

3. Visualisation en 3 dimensions

```
rglplot(g)
```

Importation des graphes

Un fichier peut être importé depuis la machine locale ou bien à travers une connexion http ou ftp. Ci-dessous deux exemples illustratifs :

- Importer un fichier GML depuis la machine locale

```
read_graph(file.choose(), format= "gml")
```

- Importer un fichier Pajek via une connexion http

```
read_graph("http://cneurocvs.rmki.kfki.hu/igraph/karate.net",
format = "pajek")
```

Exportation des fichiers

L'exportation des graphes peut être faite de deux manières :

1. Exportation des données du graphe

```
write_graph(g, file = "myGgraph.gml", format = "gml")
```

2. Exportation de la représentation du graphe sous format d'image ou de fichier PDF, etc. Pour ce faire, nous distinguons deux méthodes :
   - Soit, nous orientons la sortie du plot vers un fichier

```
jpeg("myGraph.jpg")
plot(lesmis)
dev.off()
```

   - Ou bien, nous visualisons le graphe via plot. Ensuite, nous copions la sortie du plot et nous la sauvegardons dans un fichier

```
plot(lesmis)
dev.copy(jpeg,"myGraph.jpg")
dev.off()
```

Formats des fichiers supportés

Ci-dessous une comparaison de quelques formats des fichiers supportés par iGraph :

Format	Graphe Orienté	Graphe pondéré	Attributs de nœuds/liens	Appellation des attributs	Réseau dynamique
Edge list	✓	×	×	×	×
GML	✓	✓	×	✓	×
Pajek	✓	✓	✓	×	✓
GraphML	✓	✓	✓	✓	×
CSV	✓	✓	✓	✓	✓

Table: Comparaison de quelques formats supportés par iGraph.

Exemple du format Pajek

Pajek

```
*Vertices 6
1 "Ahmed"
2 "Sarr"
3 "Boly"
4 "Sene"
5 "Amel"
*Arcslist
1 4 3 5
2 1 3 4 5
3 2 5
4 1 3
5 2 4
```

Exemple du format GraphML

GraphML

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
        <graph id="G1" edgedefault="undirected">
                <node name="Sene"/>
                <node name="Idy"/>
                <edge id=1 source="Sene" target="Idy"/>
        </graph>
</graphml>
```

Mesures locales et globales

iGraph implémente un grand nombre de mesures dont nous citons :

Mesure	Fonction iGraph
Centralité de degré	`degree`
Centralité d'intermédiarité	`betweenness`
Centralité de proximité	`closeness`
Centralité spectrale	`eigen_centrality`
Coefficient de clustering	`transitivity`
Densité	`edge_density`
Diamètre	`diameter`
Distribution des degrés	`degree_distribution`

Partie I (1/2)

- Le fichier Media.xlsx comprend deux feuilles correspondant aux données des nœuds/liens. Importez-le de sorte que le graphe résultant soit simple ;
- Visualisez le graphe sur la même fenêtre en utilisant 9 layouts différents ;
- Quels sont les types des médias qui ont plus de 35 suiveurs ?
- Quels sont les médias dont le nombre de médias qui les ont cités égal au nombre de médias qu'ils ont cités ;
- Quel est le nombre de médias dont le nombre de fois où ils étaient mentionnés égal au nombre de fois où ils ont mentionné ? Colorez ces médias en bleu et les autres en rose ;
- Affichez et interprétez l'histogramme des centralités d'intermédiarité ;

Partie I (2/2)

- Quelle lecture adoptez-vous pour la distribution des degrés ?
- Colorez en vert le plus court chemin entre *FOX News* et *WashingtonPost.com* ;
- Exportez le graphe sous format GraphML ;
- Créez de nouveau le graphe à partir du fichier GraphML :
- Vérifiez que le graphe importé du fichier GraphML est identique au graphe importé du fichier Excel ;
- Pour chaque type de mention, extrayez le graphe correspondant ;
- En utilisant le mode interactif, visualisez le graphe qui correspond au type *hyperlink*. Chaque lien bidirectionnel doit être représenté par deux arcs différents (de deux sens).

Partie II (1/2)

- Créez un graphe étoilé composé de 20 nœuds. Les liens du graphe doivent être sortants de l'ego ;
- Supprimez tous les liens adjacents à l'ego ;
- Reconstruisez de nouveau les liens du graphe en gardant le même nombre des liens du graphe du départ. La probabilité p que deux nœuds soient liés est égale à 0.7. Sachant que les boucles et les liens multiples sont autorisés ;
- Le graphe résultant est-il simple ? Si oui, pourquoi ? Si non, nettoyez-le afin qu'il soit simple ;
- Quel est l'impact de la probabilité p sur la structure du nouveau graphe par rapport au graphe du départ ?

Partie II (2/2)

- A partir de la matrice des poids, écrivez une fonction R permettant de calculer le degré pondéré des nœuds. Ensuite, déduisez comment calculer la centralité de degré ;
- Supprimez l'attribut `poids` des liens. Vérifiez que les liens ne sont plus pondérés ;
- Extrayez la matrice des plus courts chemins des nœuds du graphe. Quelles sont les valeurs possibles qu'on peut avoir sur la diagonale ?
- A l'aide de la matrice précédente, écrivez une fonction R qui permet de calculer la centralité de proximité des nœuds ;
- Vérifiez que les résultats fournis par vos fonctions sont identiques avec ceux donnés par les fonctions de iGraph.