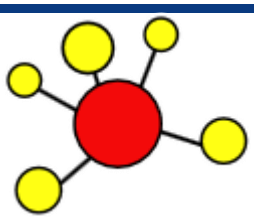
iGraph



https://www.igraph.org

Librairie de manipulation de graphes

Système d'exploitation :	Multi-plateforme
Développé par :	Gábor Csárdi and Tamás Nepusz
GNU GPL 2	Personne de contact : Vandy.Berten@Smals.be

Fonctionnalités

iGraph est une librairie de manipulation de graphes (ou réseaux), structures de données permettant de représenter des relations entre des entités. Elle est écrite en C, et disponible pour C/C++, Python, R et Mathematica. Elle est très populaire dans la communauté scientifique et des datascientists, et dispose d'un grand nombre de fonctionnalités. On y retrouve :

- Génération : de nombreux modèles de génération de graphes aléatoires ou déterministes (Barabasi, Erdos-Renyi, Full, K-regular, Tree…), très utiles soit en phase de test, soit pour comparer avec des données de production.
- Analyse : divers calculs de centralité, recherche de chemin, détection de communauté, recherche de motifs, cliques, comparaison isomorphique…
- Transformation : projection bipartite, *spanning tree*…
- Filtrage : extraction, sur base de propriétés (p.ex. degré) ou attributs.
- Visualisation : nombreux algorithmes de spatialisation (*layout, force-directed* ou déterministe) ; génération d'images dans divers formats, avec beaucoup de paramétrages disponibles (taille, couleur, forme des nœuds, courbure, épaisseur ou motif des arcs, position et police des labels).
- Import/Export dans de nombreux formats standards (graphml, gml, pajek, svg…).

iGraph peut s'utiliser en mode interactif (dans un *notebook*) ou classique (séquentiel). La création de nœuds (*vertices*), relations (*edges*) et des attributs associés se fait aisément à partir de données préparées, issues de bases de données, de dataframes ou de fichiers plats.

Conclusions & Recommandations

Dans un environnement classique (non big-data) et orienté programmation, iGraph est un incontournable pour la réalisation de « *graph/network analytics* ». Très performant, il permet l'ingestion et des analyses poussées de grands volumes de données. D'usage simple, iGraph est néanmoins très complet : une grande part de ce que la théorie des graphes connait à l'heure actuelle y est implémentée. La documentation est par ailleurs claire et très riche.

Tests & Résultats

Pour un développeur ou un datascientist familier des environnements R/Python, igraph est facile à prendre en main et est très performant, grâce à son code natif. Sa grande communauté d'utilisateurs permet de facilement trouver des réponses aux questions qu'on l'on pourrait se poser, en complément avec une documentation structurée très fournie, et nombreux tutoriaux disponibles sur le web, tant pour R que pour Python.

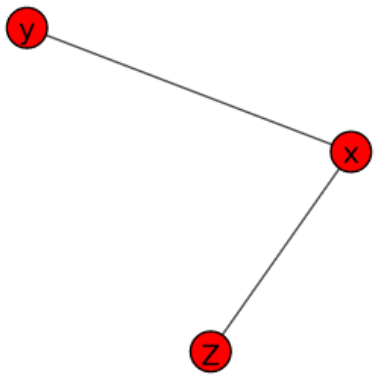
```
g = igraph.Graph()
g.add_vertices(["a","b","c"])
g.add_edges([("a", "b"),
        ("a", "c")])

g.vs["label"] = ["x","y","z"]

g.vs[2]["label"] = "Z"

igraph.plot(g)
```

Les quelques lignes de codes (en Python puis en R) ci-contre permettent de créer un graphe contenant 3 sommets (*vertices*) « a », « b » et « c », et les relations « a→b » et « a→c ». Un attribut « label » est ensuite ajouté, avec pour valeur « x » (pour le nœud « a »), « y » (pour le nœud « b ») et « z » (pour « c »). L'attribut « label » du nœud d'indice « 2 » (soit, « z ») est ensuite modifié pour recevoir un « Z ». La fonction « plot(g) » permet d'afficher (en mode interactif) le graphe, pour voir apparaître le résultat ci-contre.



L'inconvénient principal d'igraph est d'être *mono-thread*, ne permettant pas de profiter facilement de toute la puissance des processeurs actuels, pourvu en général d'au moins 4 cœurs. Certains algorithmes peuvent cependant se faire sur des parties d'un graphe, ce qui permet alors de gérer la parallélisation au niveau de R ou Python.

```
g <- graph(edges =
    c("a", "b", "a", "c"))

V(g)$label = c("x", "y", "z")

V(g)$label[2] = "Z"
```

Il faudra par ailleurs être attentif à la complexité intrinsèque des algorithmes utilisés : si l'on peut appliquer l'algorithme de Louvain (détection de communautés) en quelques secondes sur un graphe d'un million de nœuds, un algorithme tels que le calcul de « *betweenness centrality* », nécessitant de calculer le plus court chemin entre chaque paire de nœuds, demandera un temps totalement prohibitif sur le même graphe.

Pour une visualisation des graphes plus interactive, il conviendra d'exporter le graphe dans un format qui pourra ensuite être lu par un logiciel tel que Gephi[1] ou Cytoscape.

[1] : Quick Review Gephi : https://www.smalsresearch.be/publications/document/?docid=118

Conditions d'utilisation & Budget

Igraph est une librairie gratuite, sous licence GNU GPL2.