Théorie des réseaux

Chapitre 16 : Mise en œuvre sur R et Python

Jaouad Madkour

Faculté d'Économie et de Gestion de Tanger · 27 août 2026

Plan du chapitre

Sur R

Construire le réseau du cours

igraph

```
library(igraph)
g <- graph_from_literal(1-2, 1-3, 2-3, 3-4,
                        4-5, 5-6, 5-7, 6-7)
vcount(g); ecount(g)          # 7 et 8
```

Les mesures du chapitre 4

```
degree(g)                     # 2 2 3 2 3 2 2
closeness(g, normalized = TRUE)
betweenness(g, normalized = TRUE)
eigen_centrality(g)$vector
```

Les mesures du chapitre 6

```
edge_density(g)                # 0.381
transitivity(g, type = "global")   # 0.545
transitivity(g, type = "average")  # 0.667
diameter(g); mean_distance(g)      # 4 et 2.190
```

Le chapitre 10

```
cl <- cluster_louvain(g)
membership(cl); modularity(cl)
articulation_points(g)         # 3, 4, 5
```

Sur Python

Les équivalents

networkx

```
import networkx as nx
G = nx.Graph([(1,2),(1,3),(2,3),(3,4),
              (4,5),(5,6),(5,7),(6,7)])
nx.degree_centrality(G)
nx.betweenness_centrality(G)
nx.eigenvector_centrality(G)
```

La correspondance des commandes

R (igraph)	Python (networkx)
`degree`	`degree_centrality`
`betweenness`	`betweenness_centrality`
`transitivity(type="global")`	`transitivity`
`transitivity(type="average")`	`average_clustering`
`cluster_louvain`	`community.louvain_communities`

La question de la taille

Quel outil pour quelle échelle

Nœuds	Outil
$< 10^5$	`networkx`, `igraph`
10^5 à 10^7	`igraph` en C, `graph-tool`
$> 10^7$	**SNAP**, calcul distribué

Le point de rupture

`networkx` stocke chaque nœud comme un objet Python : la mémoire explose vers 10^5 nœuds.

Le passage à `igraph` ou `graph-tool`, qui stockent en C, gagne un ordre de grandeur sans changer une ligne de logique.

Les pièges

Ceux que le logiciel ne signale pas

Quatre erreurs silencieuses

Le graphe non connexe — `closeness` renvoie une valeur trompeuse au lieu d'un avertissement.

La normalisation — igraph et networkx ne normalisent pas par défaut de la même façon.

Les boucles et multi-arêtes — elles faussent degré et clustering sans message.

L'orientation perdue — importer un fichier orienté comme non orienté double silencieusement les degrés.

Le réflexe qui les attrape tous

Vérifier sur un **petit réseau connu** : celui du cours, dont on connaît les sept degrés, les distances et $\lambda_1 \approx 2{,}343$.

Si le logiciel ne redonne pas ces nombres, ce n'est pas le réseau qu'on croit.

Ce que le logiciel ne fait pas

Trois questions qui restent au chercheur

Choisir la centralité d'après le mécanisme économique, pas d'après la commodité.

Décider si le réseau est exogène — aucun package ne répond au chapitre 13.

Comparer à un modèle nul : un clustering de $0{,}45$ n'est élevé que par rapport à ce qu'un hasard à degrés conservés produirait.

Le modèle nul, en pratique

```
nul <- rewire(g, keeping_degseq(niter = 1000))
transitivity(nul, type = "global")
```

Répéter mille fois et comparer la valeur observée à la distribution obtenue.

Ce qu’il faut retenir

Le résumé du chapitre

Six points

1. `igraph` sur R, `networkx` sur Python : mêmes mesures, noms différents.
2. `networkx` cède vers 10^5 nœuds ; `igraph` et `graph-tool` gagnent un ordre de grandeur.
3. Quatre pièges silencieux : **non-connexité, normalisation, multi-arêtes, orientation perdue**.
4. Le contrôle : **retrouver les nombres d'un petit réseau connu**.
5. Le choix de la centralité relève du **mécanisme économique**, jamais du logiciel.
6. Une mesure de structure ne se lit que **comparée à un modèle nul** à degrés conservés.

Pour finir

Seize chapitres, une thèse : **la position est une variable économique.**

Elle explique un pouvoir sans propriété, une contagion sans exposition directe, une volatilité que la loi des grands nombres interdisait. **Rien de tout cela ne se voit sans le réseau.**