PROJET
UE Méthodes de ranking et recommandations
Sujet 5 : Simulation d'un Google Bombing

Maxime Gonthier - Laureline Martin

17 avril 2019

Table des matières

1 Introduction

2 Manuel utilisateur

3 Plan

Dans un premier temps nous allons effectué des tests simples en utilisant quatre type de structures différentes ainsi que trois cibles de pertinences différentes. Nous en déduirons des hypothèses sur l'eficacité de chaque structure et l'impact de la pertinence de la cible sur cette même efficacité. Dans un second temps nous étudierons l'impact sur la pertinence de graphes générés en nombre aléatoire pour étudier qu'elle structure est globalement la plus efficace. Puis nous testerons la meme choses avec cette fois ci un graphe dont les arcs le reliant a lui meme sont générées aléatoirement. Enfin nous ferons varier le nombre de sommets du graphe ataquant pour en déduire son efficacité et nous conclurons.

4 I. Tests initiaux et hypothèses

L'algorithme power calculant les pertinences est contenu dans le fichier *ranking.c*. Il n'est pas détaillé dans ce rapport mais le fichier est commenté. On considère ici le graphe du web Stanford.txt. Ce graphe est modifié par l'ajout de sommets et d'arcs afin d'augmenter la valeur d'un sommet ciblé. On considère que :

- Les sommets représentent les pages du web.
- Les arcs représentent les liens dirigeant vers d'autres pages.
- Les valeurs des sommets représentent les pertinences calculés par l'algorithme pagerank.
- Le sommet cible représente la page dont on souhaite augmenter la pertinence.
- 100 sommets sont ajoutés pour chaque test.

4.1 Explication du code

4.2 Résultats des tests initiaux

4.2.1 Stanford.txt sans modification

281903 pages
2312497 liens
132 itérations
27.627466 secondes

Voici les pertinences de base des trois sommets étudiés :
Pertinence forte : Page 280545 - 9.96199e-05
Pertinence moyenne : Page 281466 - 7.53954e-06
Pertinence faible : Page 281574 - 6.05222e-07

4.2.2 Résultats

Pertinences des sommet cible	Forte	Moyenne	Faible
Sans modification	9,96E-05	7,54E-06	6,05E-07
Difference sans modifcation	0,00E+00	0,00E+00	0,00E+00
Attaque a 100 sommets seuls	1,82E-04	5,89E-05	5,19E-05
Differences avec 100 sommets seuls	8,25E-05	5,13E-05	5,13E-05
Attaque avec un anneau de 100 sommets	1,71E-04	5,22E-05	4,52E-05
Differences avec un anneau a 100 sommets	7,17E-05	4,46E-05	4,46E-05
Attaque avec un graphe complet de 100 sommets	1,05E-04	1,08E-05	3,84E-06
Differences avec un graphe complet a 100 sommets	5,17E-06	3,23E-06	3,24E-06
Attaque avec un arbre de 100 sommets	1,72E-04	5,25E-05	4,55E-05
Differences avec un arbre a 100 sommets	7,22E-05	4,49E-05	4,49E-05
Attaque avec un arbre a attaquant unique de 100 sommets	1,38E-04	3,17E-05	2,47E-05
Differences avec un arbre a attaquant seul a 100 sommets	3,88E-05	2,41E-05	2,41E-05

4.2.3 Analyses

4.2.4 Hypothèses

4.3 Conclusion des tests initiaux et hypothèses

5 Tests avec des graphes générés aléatoirement

Nous allons désormais insérer des graphes générés aléatoirement. Ce qui est aléatoire n'est pas la structure des graphes mais le nombre de garphes générés. L'objectif de cette démarche est de déterminer quel structure est la plus efficace globalement. C'est a dire quelle structure influe le plus sur la pertinence quelle que soit la situation. De plus les resultats nous aiderons aussi a determiner l'impact qu'a une certaine structure sur la cible de manière plus générale que dans les cas prédéfinis de la partie précedente. Le nombre de sommet des graphes ajoutées est fixé a 100. La cible est également fixé ainsi que la structure des graphes ajoutées. On va par exemple insérez 5 graphes complet de nombre de sommet respectivement : 25, 10, 5, 6 et 4. Tous reliées au sommet cible. Dans un premier temps nous allons expliquer le code derrière cette démarche puis nous analyserons les résultats.

5.1 Explication du code

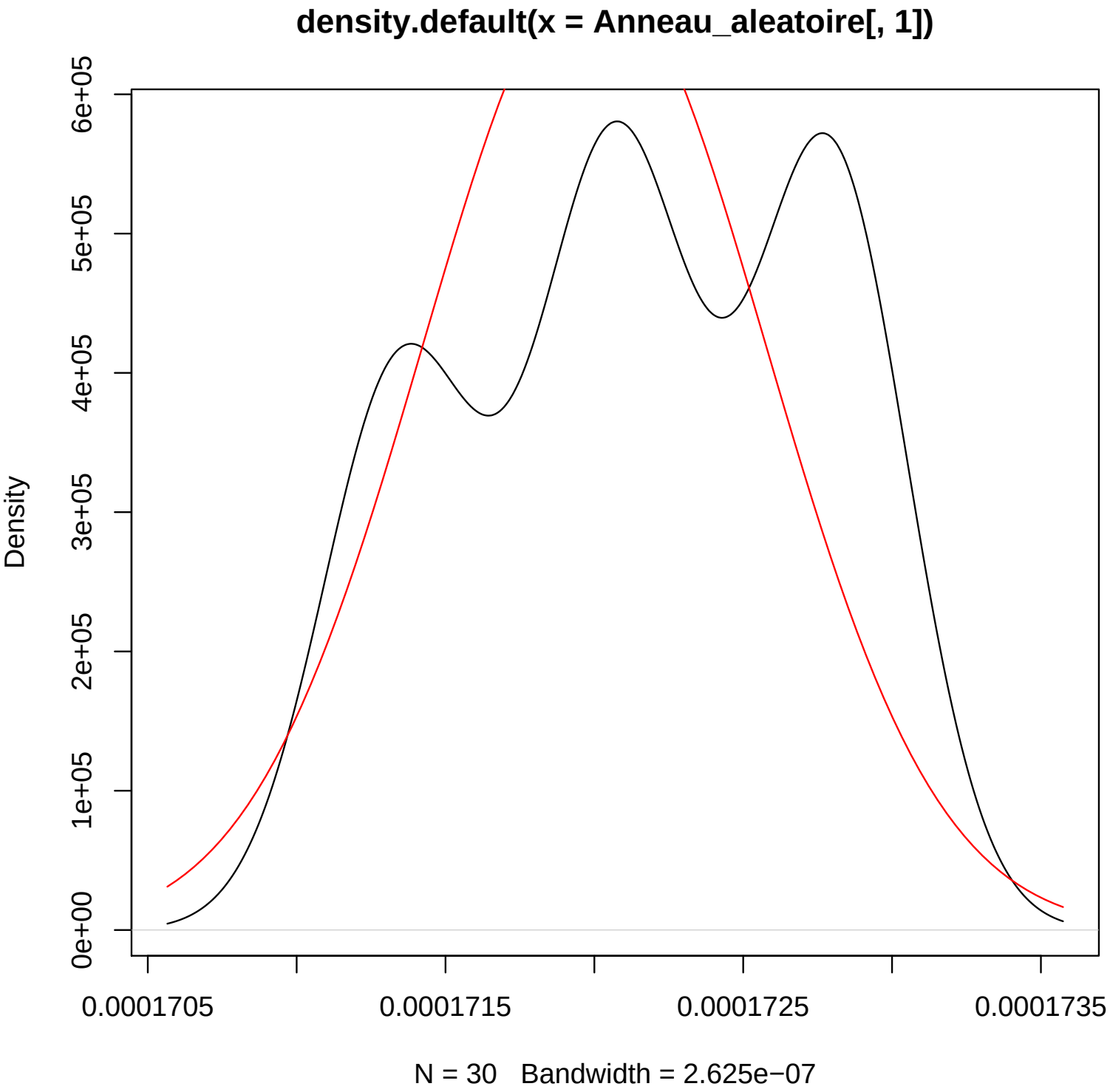
Tous est dans le fichier *ajoutsommetsattanquants.c*. Les fonctions utilisées sont *ajoutanneaualeatoire*, *ajoutcompletaleatoire* et *ajoutarbrealeatoire*. Ces fonctions reprennent en partie le code des trois fonctions presque eponyme décrite précedemment. Regardons ce qui a changé.

```
while(nbsommetrestant > 0) {
        nbajout = rand()%(100-nouveausommets);
        if (nbajout <= 3) { nbajout = 3; }
        nbsommetrestant -= nbajout;
```

nbajout représente le nombre de sommet que l'on va ajouter dans la première structure créer. Il choisis donc un nombre aléatoire entre 0 et 100 car le nombre de sommet total est limité a 100. Si le nombre choisis est inférieur a 3 on le fixe a 3 car créer des anneau ou des graphes complets de tailles inférieure a 3 reviens juste a créer des sommets seuls. *nbajout* est enlevé au nombre de sommet restant a ajouter. Puis on lance la construction de la structure comme vu précedemment avec *nbajout* représentant le nombre de sommets. A la fin de cette itération *nbajout* reprends un nombre aléatoire qui cette fois prend une valeur entre 0 et 100 moins le nombre de sommets ajouté precedement représenté par *nouveausommets*.

5.2 Résultats des tests

5.3 Analyse graphes en anneau


density.default(x = Anneau_aleatoire[, 1])
Density
0e+00
1e+05
2e+05
3e+05
4e+05
5e+05
6e+05
0.0001705
0.0001715
0.0001725
0.0001735
N = 30 Bandwidth = 2.625e−07