Classification spectrale de graphes

Thomas Soubiran
Ceraps (UMR 8026 CNRS - Université de Lille)

Frognet

Toulouse, 28 mai 2019

Introduction

- **objet de la communication :** analyse **spectrale** de graphes
  - plus particulièrement, **le laplacien du graphe**
    - avec une présentation de différentes interprétations

      optimisation de la coupe du graphe, marches aléatoires, . . .
    - entre autres possibilités

      blockmodel stochastique, analyse des correspondances, . . .
    - et en lien avec d'autres applications de l'analyse spectrale à des graphes
  - avec une application aux flux entre **commune de résidence** et **commune du lieu de travail**

    issus du recensement de populations
- travail **en cours**
- de façon concomitante, aperçu de l'utilité de **l'algèbre linéaire** pour travailler avec des (grands) graphes

  pour stocker, manipuler, chercher, analyser, . . .

Présentation des données

Le recensement général de population

- les volumes des trajets inter-communaux sont issus **du recensement de la population**
- bon an, mal an, le recensement s'est déroulé en France **tous les cinq ans** de 1801 jusqu'aux années cinquante
- l'intervalle inter-censitaire a ensuite progressivement **augmenté**
- jusqu'à atteindre **neuf ans** en 1999
- l'INSEE a ensuite procédé à une **« rénovation »** radicale du RGP

  et ce, à *l'invitation* des pouvoirs publics

- qui a été effective **partir de 2004**

Les enquêtes annuelles de recensement de la population

- enquête tournante **par sondage** réalisée annuellement sur une période de six ans
- durant laquelle seule **une fraction** des ménages (logements ?) est interrogée
- en effet, chaque année :
  - seule une fraction des communes de moins de 10 000 hab.
  - seule une fraction des ménages des communes de plus de 10 000 hab.
  - sont interrogées
- ce changement a suscité de **nombreuses interrogations** dès sa mise en œuvre

  et continue d'en susciter, particulièrement de la part d'élus locaux
- les flux ne sont donc pas **comptabilisés** mais bien **inférés**

Note : pour plus de détails, voire notamment le n° spécial d'*Économie et statistiques* de 2016 sur le recensement rénové (n° 483-484-485 – 2016)

Les trajets entre communes

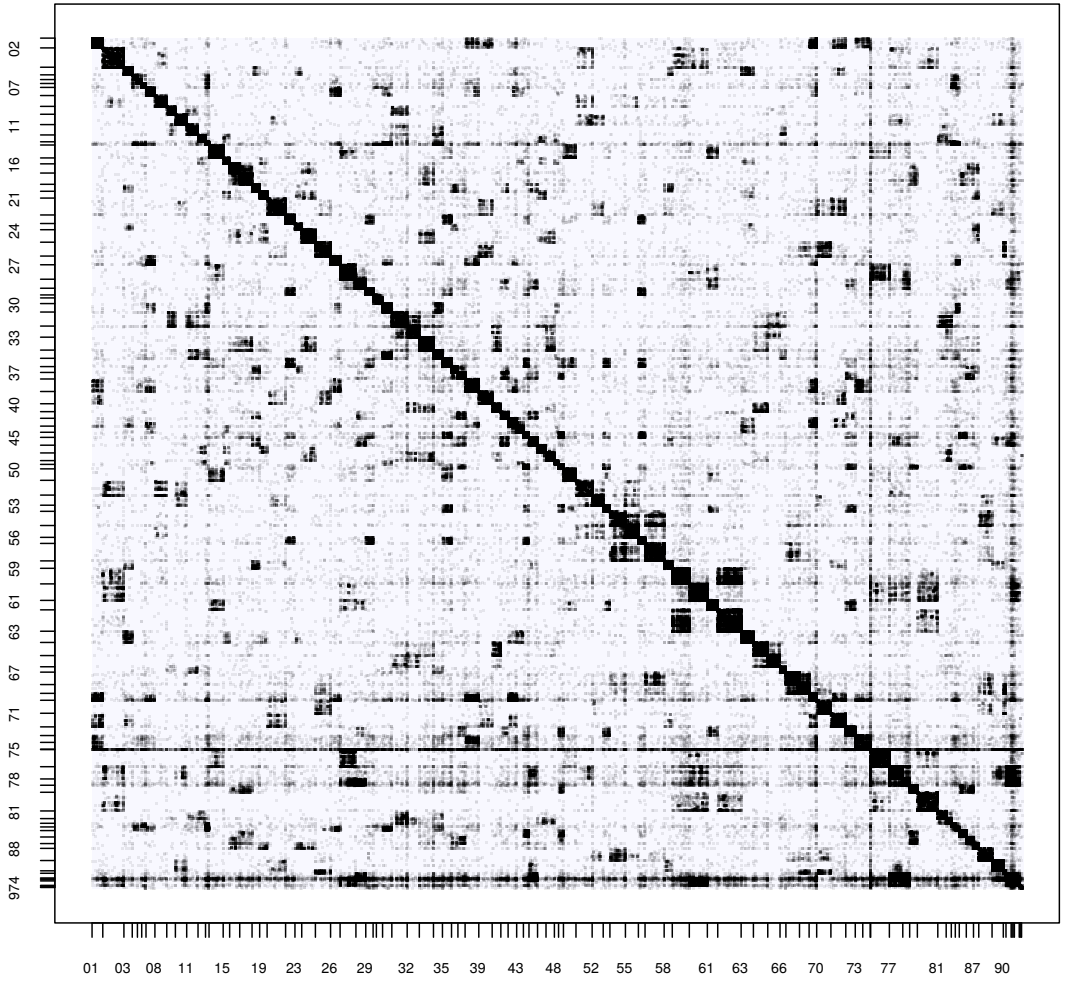


FIGURE 1 – Relations binaires entre les communes ($a_{ij} > 0$)), tri par cantons

Haute-Garonne

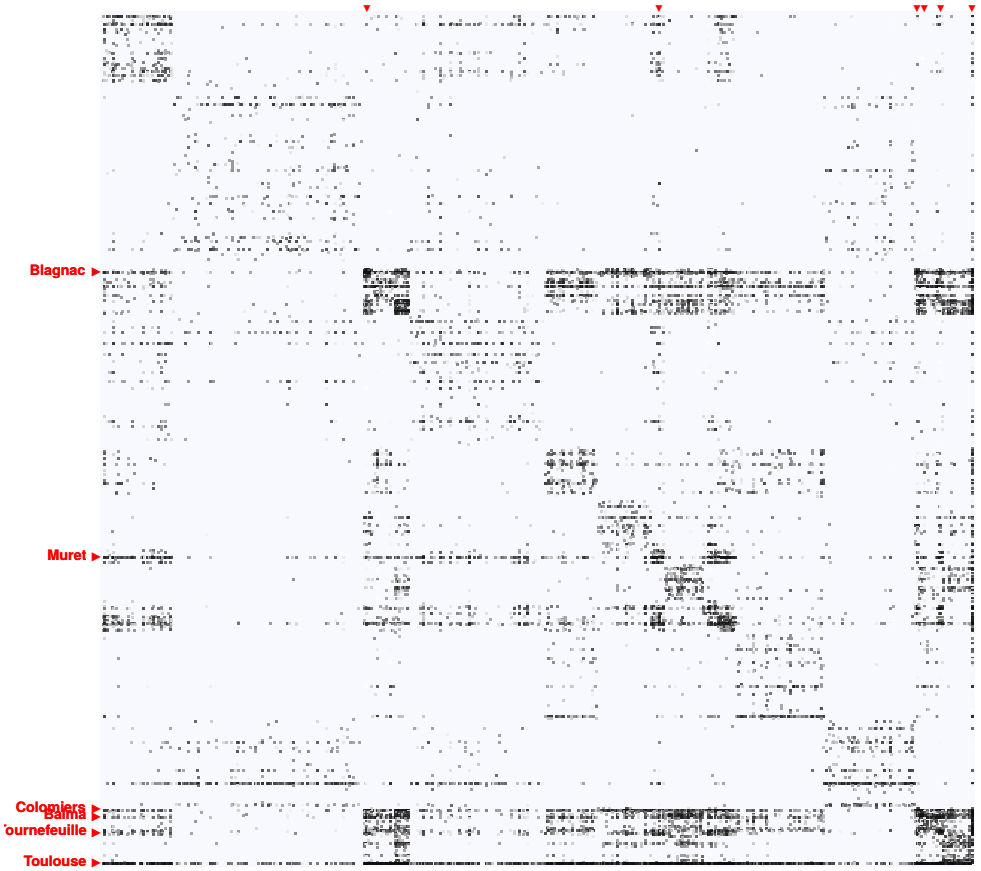


FIGURE 2 – Flux entre les communes de la Haute-Garonne, tri par cantons

La structure sous-jacente du graphe

- les deux graphes permettent de faire ressortir la structure **territoriale** des flux
- autre **topologie** : la géographie
- référence qui permet **d'évaluer** les analyses

  avec d'autres critères que les seuls mesures statistiques

- utile pour **des tests**
- avant application à **un autre graphe** : **les signatures** à un site **de pétitionnement en ligne**

La base `lapetition.be`

- site de pétitionnement lancé **fin 2006**
- extraction de la base `SQL` servant de *backend* au site en **février 2015**
- la base renseigne
  - plus de **15 000 pétitions**

    en fait, plutôt 12 4000, le nombre variant en fonction de la table considérée. . .
  - près de **3,8 millions** de signatures
- la base **renseigne aussi** (avec un nombre de données manquantes variables) :
  - `hash` des adresses électroniques utilisées pour valider les signatures

    ces `hashs` *pouvant correspondre à plusieurs personnes utilisant l'adresse*
  - prénom du signataire
  - horodatage de la signature et de sa validation
  - la commune de résidence du signataire
  - . . .

Les co-signatures aux pétitions

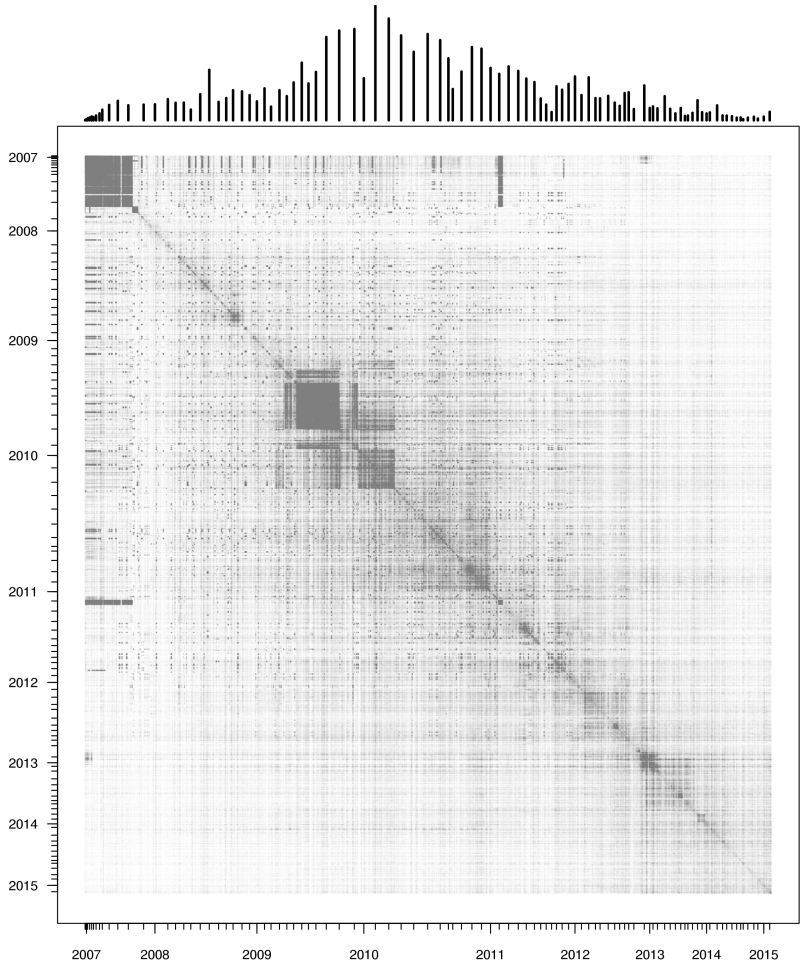


FIGURE 3 – Relations binaires entre les pétitions ($n_{ij} > 0$), tri par date de création des pétitions

L'effet de plate-forme

- le graphe permet de faire ressortir la structure avant tout **temporelle** qui relie les pétitions
  - en effet 21 % signatures multiples ont lieu **dans l'heure** suivant la précédente signature

    tendance sans doute renforcée par la mise en avant sur le site des pétitions les plus signées du moment
  - une large partie des liens entre les pétitions (et donc les signataires) provient très probablement de **la navigation sur le site**

    la signature de pétitions en ligne conduit donc à la signature de pétitions que les personnes n'auraient peut-être jamais signées autrement
  - à ce titre, il semble de plus que les comportements changent avec **la pratique du site**

    plus le nombre de signatures augmente, plus le temps entre deux signatures se réduit
- illustration de **la dualité** des graphes biparti
- qui, combiné à **l'éparpillement des signatures**
- et complique l'analyse du graphe

Les flux entre communes

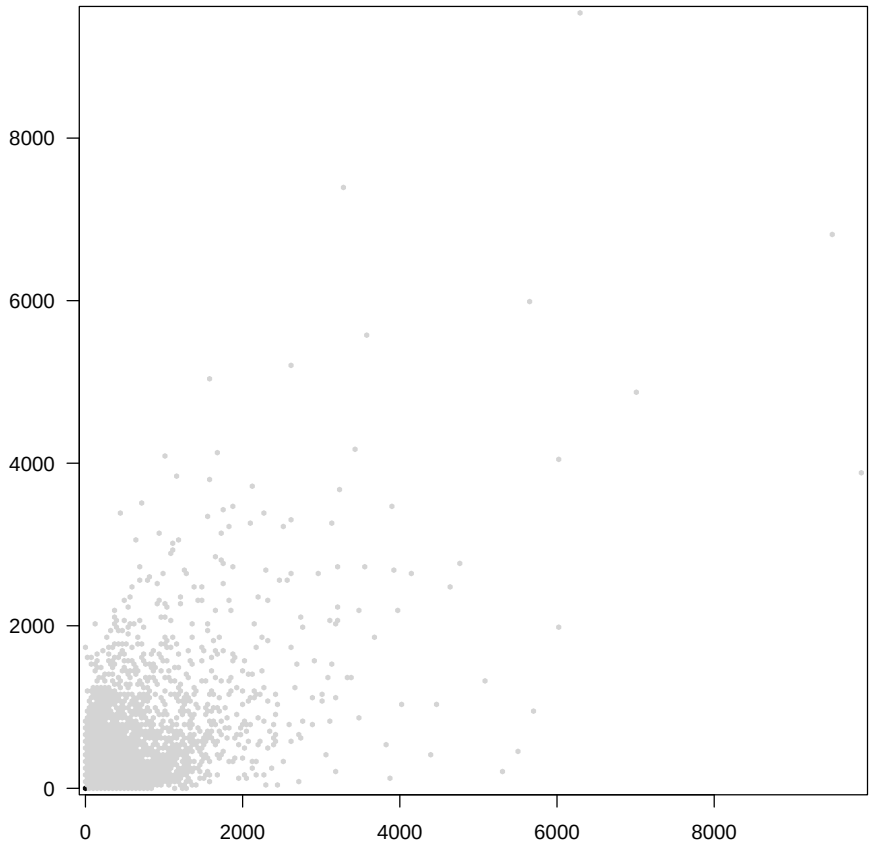


FIGURE 4 – Nuage de points des flux sortant et rentrants

Enfin

- le graphe des flux est **orienté** : flux rentrants et flux sortants
- les flux ont été **symétrisés** ($a_{ij}^{sym} = a_{ij} + a_{ji}$)

  Note : il existe des variantes pour graphe orienté de l'analyse spectrale cf. infra)

- de plus,
  - les analyses portent sur **la métropole**
  - **les arrondissements** de Paris, Lyon et Marseille n'ont pas été agrégés

Graphes et algèbre linéaire

Analyse spectrale de graphe

- avec les moindres carrés, **la décomposition en valeurs propres** est la cheville ouvrière de l'analyse de données

  et ce, tant d'un point de vue théorique que pratique

- l'analyse de graphes ne fait **pas exception**
- il existe **une relation privilégiée** entre analyse de graphe et analyse spectrale

  - les valeurs propres **encodent** en effet différentes propriétés des graphes
  - notamment liées à **la conductivité** du graphe

Décomposition en valeurs propres

- pour une matrice $\mathbf{A}$
- $\lambda \in \mathbb{C}$ est **une valeur propre** de $\mathbf{A}$
- s'il existe un vecteur $\mathbf{x} \in \mathbb{C}$ qui satisfait la relation

$$\lambda \mathbf{x} = \mathbf{A}\mathbf{x} \tag{1}$$

où $\mathbf{x}$ est **un vecteur propre**

- si $\mathbf{A}$ est **symétrique** ($\mathbf{A}s \in \mathbb{R}^{n \times n}$) :
  - les vecteurs propres propres sont réels
  - ils sont orthogonaux ($\mathbf{x}_1 \perp \mathbf{x}_2$)
  - si $\mathbf{A}$ est définie semi-positive ($\mathbf{x}^\top \mathbf{A}\mathbf{x} \geq 0$)$s$, les valeurs propres sont aussi réelles
- pour les matrices **non symétriques**, cf. décomposition en valeurs singulières

Décomposition en valeurs propres

- de plus, les valeurs propres sont **ordonnées** : $\lambda_1 \leq \ldots \leq \lambda_n$

  plusieurs valeurs propres peuvent être égales (multiplicité)

- et, entre autres choses,

$$\text{trace}(\mathbf{A}) = \sum_i^n a_{ii} = \sum_i^n \lambda_i \tag{2}$$

- **caractérisation variationnelle** des valeurs propres :

$$\lambda_1 = \min_{\mathbf{x} \neq 0} \frac{\mathbf{x}^\top \mathbf{A}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}} \tag{3}$$

$$\lambda_2 = \min_{\substack{\mathbf{x} \neq 0 \\ \mathbf{x}^\top \mathbf{x}_1 > 0}} \frac{\mathbf{x}^\top \mathbf{A}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}} \tag{4}$$

$$\ldots \tag{5}$$

Exemples d'applications à l'analyse de réseaux sociaux

- **centralité des valeurs propres** : cf. infra p.33
- **centralité de Katz** :

$$x_i = \alpha \sum_j^n a_{ij} x_{ij} + \beta \tag{6}$$

$$\mathbf{x} = \alpha \mathbf{A}^\top \mathbf{x} + \beta \mathbf{1} \tag{7}$$

- `PageRank` : cf. infra p.36
- **le laplacien du graphe** : cf. infra p.37

Algèbre linéaire

- l'application de l'algèbre linéaire aux graphes **ne se limite toutefois pas à l'analyse spectrale**
- et ce, d'un point de vue **théorique**

  étude des propriétés de certaines catégories de graphes

- mais aussi **pratique** comme le stockage et la manipulation de graphes
- ou encore l'implémentation **de primitives** comme des algorithmes de recherche

  Exemple : sle parcours en largeur (`BFS`)

- en effet, un graphe peut être représenté par **une matrice d'adjacence**

La matrice d'adjacence

- la matrice d'adjacence est **une matrice carrée** de dimension $(|V|, |V|)$:

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & \dots & a_{1,j} & \dots & a_{1,J} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{1,i} & \dots & a_{i,j} & \dots & a_{i,J} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{I,1} & \dots & a_{I,j} & \dots & a_{I,J} \end{pmatrix}$$

- $a_{i,j}$ manifestent **les liens** entre les sommets du graphe
- elles peuvent être **binaires ou valuées**
- la matrice peut être **symétrique ou asymétrique**

Matrices éparses

- les matrices d'adjacences sont généralement **éparse**

  seule une fraction de $a_{ij} > 0$

- et ça, d'autant plus que **le nombre de sommets** (dimensions de la matrice) croît

  pour le graphes des flux inter-communaux : $\frac{\#a_{ij} \,|\, a_{ij} > 0}{|V|^2} = \frac{908279}{1254363889} \simeq 7e^{-4}$

- dans ce cas de figure, le graphe peut être représenté au moyen **d'une matrice éparse**
- pour lesquels différentes **implémentations** efficaces existent
  - comme `SuiteSparse` qui est accessible dans `R` via la paquet `Matrix`
  - qui propose un large éventail de structures de données

    éparses symétriques, éparses asymétriques, diagonales, structurées, denses,. . .

  - et de méthodes pour les opérations sur ces matrices

    y compris de types différents

La création du graphe des flux

- création **d'un dictionnaire des communes**
  - en le triant sur le code commune
  - ce dictionnaire renseigne les sommets du graphe et donc les lignes (resp. les colonnes) de la matrice
- assignation des indices pour obtenir **le triplet** (i, j, n_{ij})

Départ	Destination	n
L'Abergement-Clémenciat	Bourg-en-Bresse	35
L'Abergement-Clémenciat	Châtillon-sur-Chalaronne	55
L'Abergement-Clémenciat	Feillens	5
...	...	...

i	j	n
1	49	35
1	86	55
1	146	5
...	...	...

avec la fonction `match()`

- suppression **des boucles** et des trajets vers **des destinations à l'étranger** (frontaliers)

La création du graphe (suite)

- création de **la matrice éparse**

```
1          A ← sparseMatrix(
2                 i = i
3               , j = j
4               , x = n
5               , dims = c(nrc, nrc)
6               , dimnames = list(communes0$CODGEO, communes0$CODGEO)
7          )
```

- tout ça en un plus de $\sim$ 1 **seconde**...

Note :

- `SuiteSparse` utilise une version compressée et ne stocke pas les paires i, j
- mais seulement les i
- les indices des colonnes peuvent être retrouvés en utilisateur un vecteur p de dimension $|J|$ qui stocke le nombre d'éléments de la colonne

Exemples d'opérations

- changer l'ordre des colonnes pour **trier en fonction du cantons** :

```
1   A[perm,perm]
```

avec `perm` l'indice des sommets triés par cantons

- sous-ensemble pour extraire **le super-composant**

```
1   cmpn ← graph.components(A)
2   idx  ← cmpn==1
3   A    ← A[idx,idx]
```

- la fonction `graph.components()` permet **d'attribuer chaque sommet à son composant** (sous-graphe connecté) cf. infra p.29
- cette opération est nécessaire pour l'analyse spectrale **du laplacien du graphe**

Exemples d'opérations

- **degré :**
  - sortant : $\sum_j^n a_{ij}$ `rowSums(A)`
  - rentrant : $\sum_i^n a_{ij}$ `colSums(A)`
- **symétrisation** $(\mathbf{A} + \mathbf{A}^\top)$ `A + t(A)`
- **nombre de chemins** entre sommets

Cheminements

- calcul **du nombre de chemins** entre sommets pour les trajet de longueur k
  - pour un trajet de longueur 1 **(mes amis)**, le nombre de chemins entre v_i et v_i est donné par les entrées de $\mathbf{A}$
  - pour un trajet de longueur 2 **(amis de mes amis)**, le nombre de chemins est donné par $\sum_k^N a_{ik} a_{kj}$, soit
$$\mathbf{AA} = \mathbf{A}^2 \tag{8}$$
  - pour un trajet de longueur 3 **(amis des amis de mes amis)**, le nombre de chemins est donnée par $\sum_k^N \sum_k^N a_{ik} a_{kl} a_{lj}$, soit
$$\mathbf{AAA} = \mathbf{A}^3 \tag{9}$$
  - plus généralement (amis des amis des amis. . . de mes amis),
$$\mathbf{A}^k = \mathbf{AAA} \ldots \mathbf{A} \tag{10}$$
  - la diagonale de $\mathbf{A}^k$ donne le nombre de fois où un chemins partant de v_i repasse par v_i

    les amis de mes amis sont-ils mes amis ? – cf. triangles p.31

- `Matrix` : `A^k`

  cf. infra p.31 pour une autre approche utilisant les valeurs propres

Fazit

- l'objet ainsi crée **s'intègre parfaitement** dans R
- pour réaliser **d'autres opérations**
- mais aussi pour la **sérialisation**
- **plus généralement**,
  - pas besoin de convertir un objet représentant le graphe en matrice pour certaines opérations
  - organisation des données en mémoire (localité)
  - parallélisation

    la parallélisation pour les matrices éparses reste toutefois un domaine de recherche actif (graphes hyper-épars)

Fazit

- différentes opérations sur le graphe peuvent donc être **facilement réalisées** avec les fonctions|méthodes proposées par un paquet comme `Matrix`
- l'intérêt de la représentation d'un graphe par une matrice éparse va cependant **au de-là** du stockage et la réalisation d'opérations élémentaires sur le graphe
- car
  - différents algorithmes conçus pour des graphes peuvent être **exprimés** dans le langage de l'algèbre linéaire
  - certains calculs peuvent aussi être réalisés **en reformulant** le problème du point de vue de l'algèbre linéaire

Composants du graphes

- `graph.components()` est basé sur une variante de **l'algorithme `BFS`**
- comme le graphe est représenté par une matrice éparse, l'implémentation de l'algorithme peut être **vectorisée**
  - sans utiliser '['
  - mais en accédant directement aux `slots` de la structure de données (`A@i`, `A@p`)
  - et en utilisant $\sim$ l'arithmétique des pointeurs
  - pour des raisons d'efficacité
- il peut aussi être exprimé dans **le langage de l'algèbre linéaire**
- et implémenté avec les opérations fournies par le package `Matrix`
- la version vectorisée est toutefois **un peu plus rapide**

  effet du dispatch des classes `S4` ?

Plus courts chemins

- l'algorithme BFS peut être exprimé de façon concise dans **le langage de l'algèbre linéaire**
- **Exemple :** calcul du plus court chemin :

```
1  sp.bfgs ← function(G, src=1){
2    n       ← nrow(G)
3    d       ← integer(n)
4    d[src] ← 1
5    u       ← d
6    for (i in 1:n){
7      u   ← ( G %*% u ) & (d==0)
8      idx ← as.vector(u!=0)
9      if( all(idx==F)) break
10     d[idx] ← i+1
11   }
12   d
13 }
```

- le remplacement de **d** par une matrice permet de faire la recherche **sur tous les sommets**
- **limite :** SuiteSparse n'est pas parallélisée

Triangles

- la décomposition en valeur propre peut servir **à compter le nombre de chemins**
- en effet,

$$\mathbf{A}^n\mathbf{X} = \lambda^n\mathbf{X}$$

- les valeurs propres peuvent donc aussi servir à calculer **le nombre de triangles** :
  - nombre total de triangles

$$\begin{aligned}\Delta(\mathbf{A}) &= \frac{1}{6}\text{trace}(\mathbf{A}^3) \\ &= \frac{1}{6}\sum_i^n \lambda_i^3\end{aligned}$$

  - nombre de triangles auxquels le sommet v_i participe

$$\Delta(\mathbf{A})_{v_i} = \frac{\sum_i^n \lambda_i^3 x_{ij}^2}{2}$$

Etc.

- **autres exemples :**
  - algorithme de Brandes pour la centralité d'intermédiarité
  - algèbre tropical (produit min-max)

    algorithme de Prim
  - graphes de Kronecker
  - ...
- l'utilité de l'application de l'algèbre linéaire aux graphes a conduit à **une normalisation** : `graphBLAS`

  par analogie à `BLAS`
- **plusieurs implémentations** sont en cours de développement

  dont une basée sur `SuiteSparse`

Analyse spectrale de graphe

- **centralité des valeurs propres :**

$$\lambda \mathbf{x} = \mathbf{A}\mathbf{x} \text{ (décomposition en valeur propre de } \mathbf{A})$$
$$\mathbf{x} = \lambda^{-1}\mathbf{A}\mathbf{x}$$
$$x_i = \frac{1}{\lambda}\sum_{j,j\neq i} a_{i,j}x_j$$

  - la centralité d'un sommet x_i est donc proportionnelle **à la centralité de ses voisins immédiats**

    autrement dit, les sommets les plus centraux sont ceux comptant le plus de voisins centraux eux aussi

  - mais aussi **du nombre de ses voisins**
  - cas particulier **de marche aléatoire** (*random walk*) sur un graphes

Marches aléatoires sur un graphe

- différente mesures de centralité sont liées à la notion **de marche aléatoire sur un graphe**
- l'idée est
  de faire ressortir les sommets **les plus visités** si on parcourt le graphe **de façon aléatoire** mais pas nécessairement de façon uniforme
- les probabilités d'atteindre un sommet v_i sont données par **la matrice de transition**
  - graphe non-dirigé

$$\mathbf{W} = \mathbf{D}^{-1}\mathbf{A} \text{ soit } p_{ij} = \frac{a_{ij}}{d_{v_i}} \tag{11}$$

  - graphe dirigé

$$\mathbf{A}\mathbf{D}_{ex}^{-1} \text{ sortant} \tag{12}$$

$$\mathbf{D}_{in}^{-1}\mathbf{A} \text{ rentrant} \tag{13}$$

Marches aléatoires sur un graphe

- la probabilité d'atteindre la sommet v_i à **la k étape**

$$\mathbf{p}^{(k+1)} = \sum_i w_{ij} p_i(k) = \mathbf{W}_{ex}\mathbf{p}^{(k)} = \mathbf{W}_{ex}^{(k)}\mathbf{p}^{(1)} \tag{14}$$

- sous certaines conditions, la marche aléatoire **converge vers la distribution** :

$$\lim_{k\to\infty} \mathbf{W}_{ex}\mathbf{p}^{(ks)} = \pi \tag{15}$$

- π est donc **un vecteur propre** (correspondant à une valeur propre de 1)

  car π satisfait la relation $\pi = \mathbf{W}\pi$

- **Problème :** certains sommets peuvent mener **nulle part**
  - on a donc une division par 0 dans la matrice de transition
  - **Exemple :** une page sans liens sortants

Beam me up Scotty

- **Solution :** la téléportation
- qui consiste à ajouter **une petite probabilité uniforme** d'accéder à n'importe quelle page

$$\pi = \alpha \frac{1}{n} \mathbf{1}\mathbf{1}^\top + (1 - \alpha)\mathbf{W}_{ex}\pi \tag{16}$$

- en calculant **le premier vecteur propre** de (16)
- on obtient alors le fameux v**ecteur** `PageRank`
- **Note :** pour les matrices de grande taille, on peut calculer directement le vecteur en reformulant (16)

Le laplacien du graphe

- dans les deux cas précédents, on calcule **le premier vecteur propre** de la matrice d'adjacence du graphe (**ou d'une transformation**)
- c'est aussi le cas pour **le laplacien du graphe**

  cf. `ACP`
- mais sans forcément **se limiter au premier vecteur**

  où plutôt le deuxième (vecteur de Fiedler)

Le laplacien du graphe

- le laplacien a pour **forme** :

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \text{ avec } l_{ij} = \begin{cases} d_{v_i} & \text{si } i = j \\ -a_{ij} & \text{si } i \neq j \end{cases} \tag{17}$$

où $\mathbf{D}$ est une matrice avec les degrés d_i des sommets v_i dans la diagonale

- **variante normalisée :**

$$\mathbf{L}_{\text{norm}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \tag{18}$$

avec

$$l_{ij} = \begin{cases} 1 & \text{si } i = j \\ -\frac{a_{ij}}{\sqrt{d_{v_i}}\sqrt{d_{v_j}}} & \text{si } i \neq j \end{cases}$$

Le laplacien du graphe

- **forme quadratique du laplacien :**

$$\begin{aligned}\mathbf{x}^\top \mathbf{L}\mathbf{x} &= \frac{1}{2}\sum_{i,j}^{n} a_{ij}(f_i - f_j)^2 \\ &\geq 0\end{aligned}$$

Exemple : graphe régulier

$$\mathbf{x}^\top \mathbf{L}\mathbf{x} = (x_1 x_2)\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = (x_1 - x_2)^2$$

Note : dans ce cas, $\mathbf{x} = (\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}})^\top$ est un vecteur propre de $\mathbf{L}$ et a 2 pour valeur propre

De plus, $[\mathbf{L}\mathbf{x}]_i = d_{v_i} * (x_i - \text{la moyenne des } x \text{ des voisins de } i)$

- **forme quadratique du laplacien normalisé :**

$$\begin{aligned}\mathbf{x}^\top \mathbf{L}_{\text{norm}}\mathbf{x} &= \frac{1}{2}\sum_{i,j}^{n} a_{ij}\left(\frac{f_i}{\sqrt{d_{v_i}}} - \frac{f_j}{\sqrt{d_{v_j}}}\right)^2 \\ &\geq 0\end{aligned}$$

Le laplacien du graphe

De plus,

- $\mathbf{L}$ et $\mathbf{L}_{\text{norm}}$ sont des matrices **définies positives** si $a_{ij} \geq 0$
- la plus petite valeur propre de $\mathbf{L}$ **vaut 0** et elle a $\mathbf{x} = \mathbf{1}$ **pour vecteur propre**

  pour $\mathbf{L}_{\text{norm}}$, $\mathbf{x} = \mathbf{D}^{1/2}\mathbf{1}$
- $\mathbf{L}$ et $\mathbf{L}_{\text{norm}}$ ont *n* **valeurs propres** réelles et non négatives :
  $0 = \lambda_1 < \lambda_2, \ldots, \lambda_n$
- les valeurs propres sont comprises entre **0 et 2**
- la multiplicité dus zéro donne le nombre **le nombre de composants**

Graphes à la découpe

- l'analyse du laplacian peut être vu comme **un problème de découpage de graphe**
- on souhaite **répartir les sommets dans K partitions** $A_1, \ldots, A_k, \ldots, A_K$ homogènes de façon à ce que
  - les arêtes entre groupes ont un poids faible
  - les arêtes à l'intérieur des groupes ont un poids forts

  cf. variance intra-extra
- **la coupe** (*cut*) du graphe a pour expression

$$cut(A_i, \bar{A}_i) = \sum_{v_i \in A_i, v_j \bar{A}_i} a_{ij}$$

Graphes à la découpe

- **deux critères d'homogénéité :**
  - ***Ratio cut* :**

$$RCut(A_1, \ldots, A_K) = \frac{1}{2} \sum_{k=1}^{K} \frac{cut(A_i, \bar{A}_i)}{|A_i|} \tag{19}$$

  - ***Normalized cut* :**

$$Ncut(A_1, \ldots, A_K) = \sum_{k=1}^{K} \frac{cut(A_i, \bar{A}_i)}{vol(A_i)} \tag{20}$$

    avec $vol(A_i) = \sum_{v_i \in A_i} d_i$

- le laplacien et le laplacien normalisé sont respectivement **la solution de (19) et (20)**

Marche aléatoire sur le graphe

- **marche aléatoire sur le graphe :**

$$\mathbf{W} = \mathbf{A}\mathbf{D}^{-1} \tag{21}$$

- le laplacien normalisé **est lié** à une marche aléatoire sur le graphe par

$$\begin{aligned} \mathbf{L}_{rw} &= \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{1/2} && (22)\\ &= \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} && (23)\\ &= \mathbf{I} - \mathbf{L}_{\text{norm}} && (24) \end{aligned}$$

- $\mathbf{L}_{rw}$ et $\mathbf{L}_{rw}$ ont (presque) **les mêmes valeurs propres** $\lambda_i^{rw} = 1 - \lambda_i^{norm}$

  toutefois, on a $1 = \lambda_i^{rw} > \ldots > \lambda_n^{rw}$

- et leurs vecteurs propres **sont identiques**

Classification spectrale de graphe

En pratique, **la classification spectrale** consiste à :

1. transformer la matrice d'adjacence $\mathbf{A}$

   avec (18), (18) ou (24)

2. calculer les k plus grandes|petites valeurs propres et vecteurs correspondants $\mathbf{X}$
3. réaliser une classification sur les $\mathbf{X}$

   comme *k-means*

Application

Valeurs propres

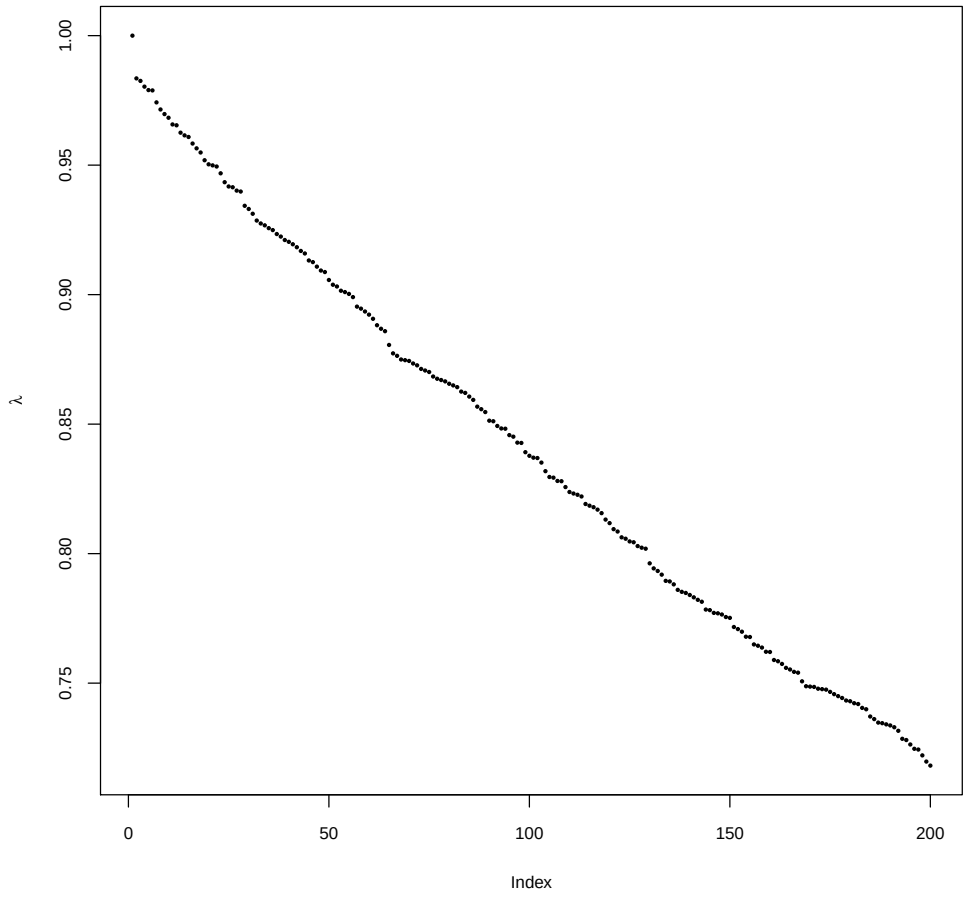
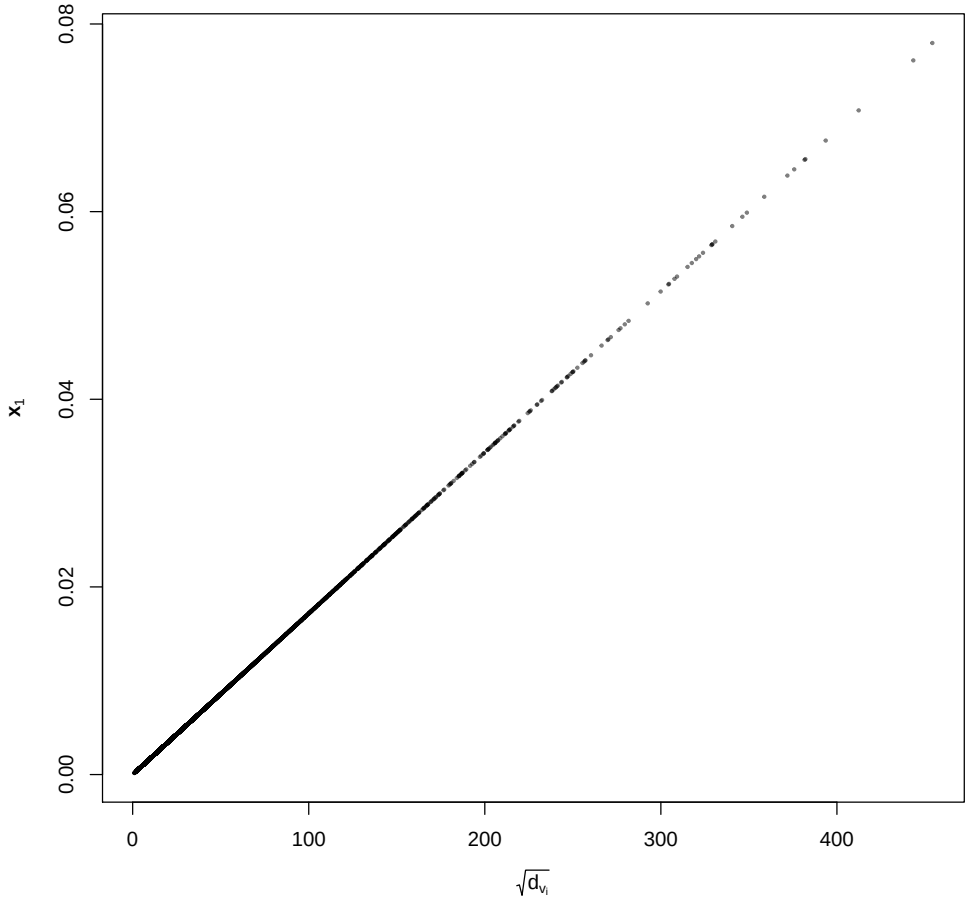
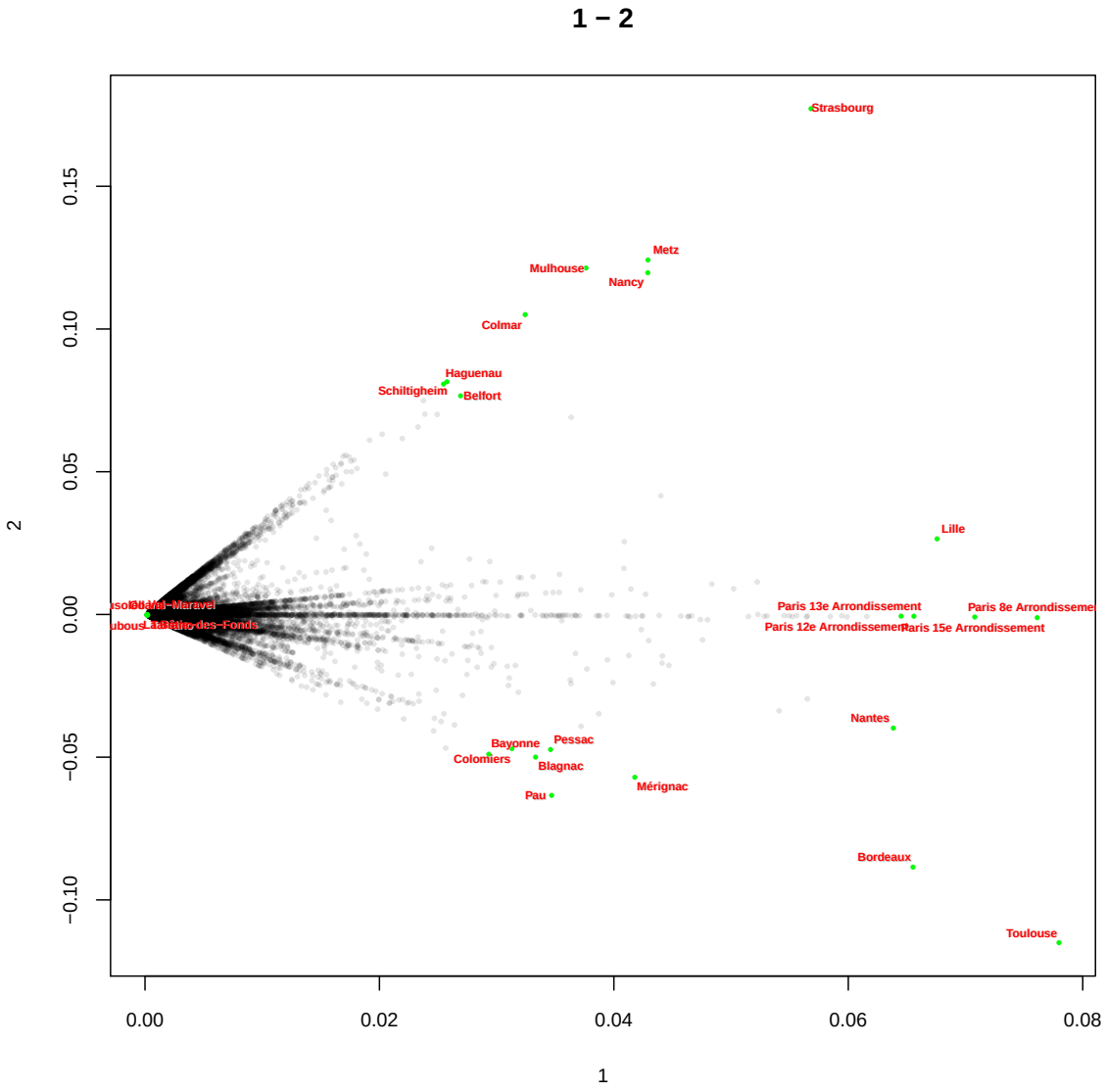
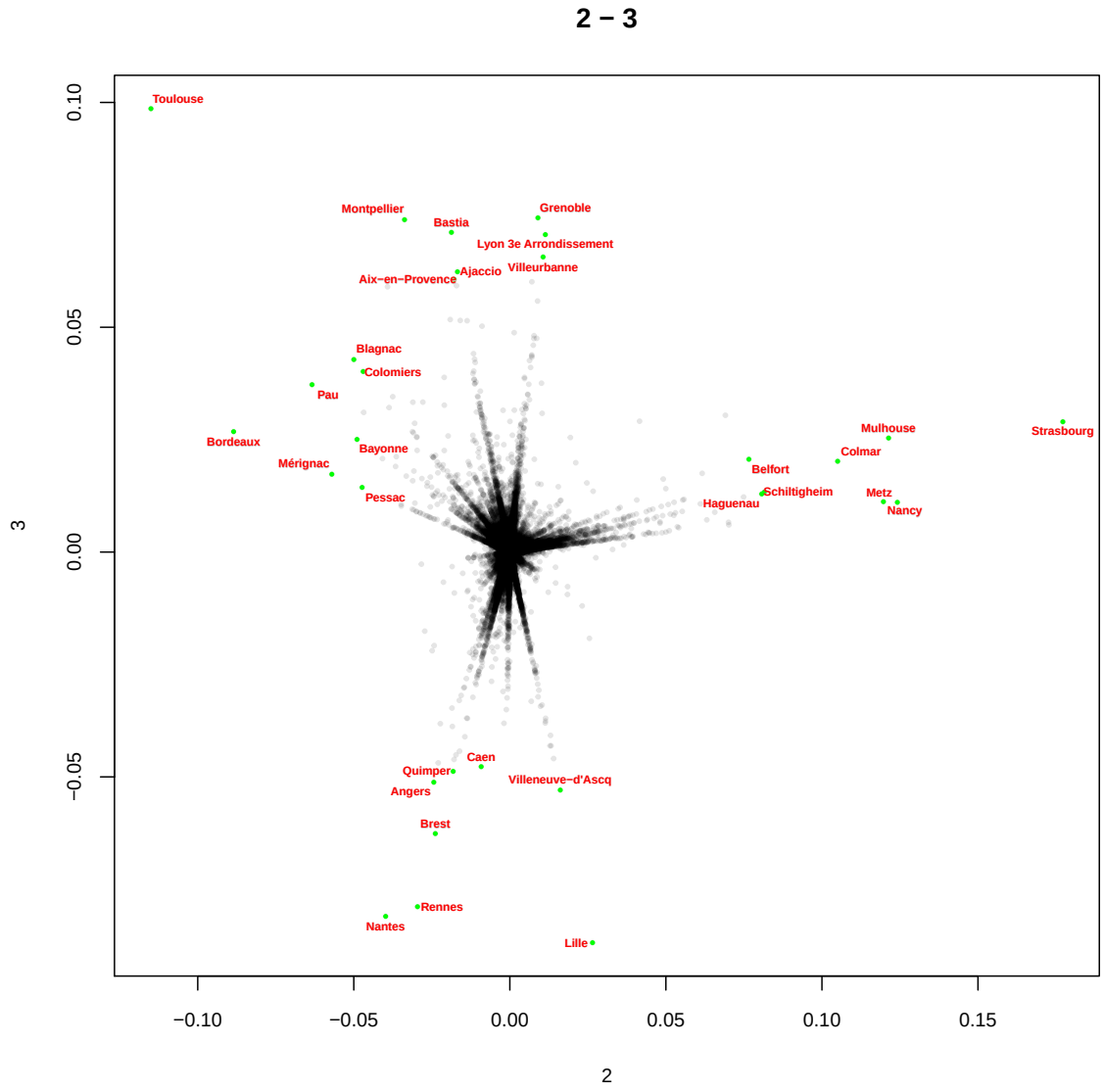


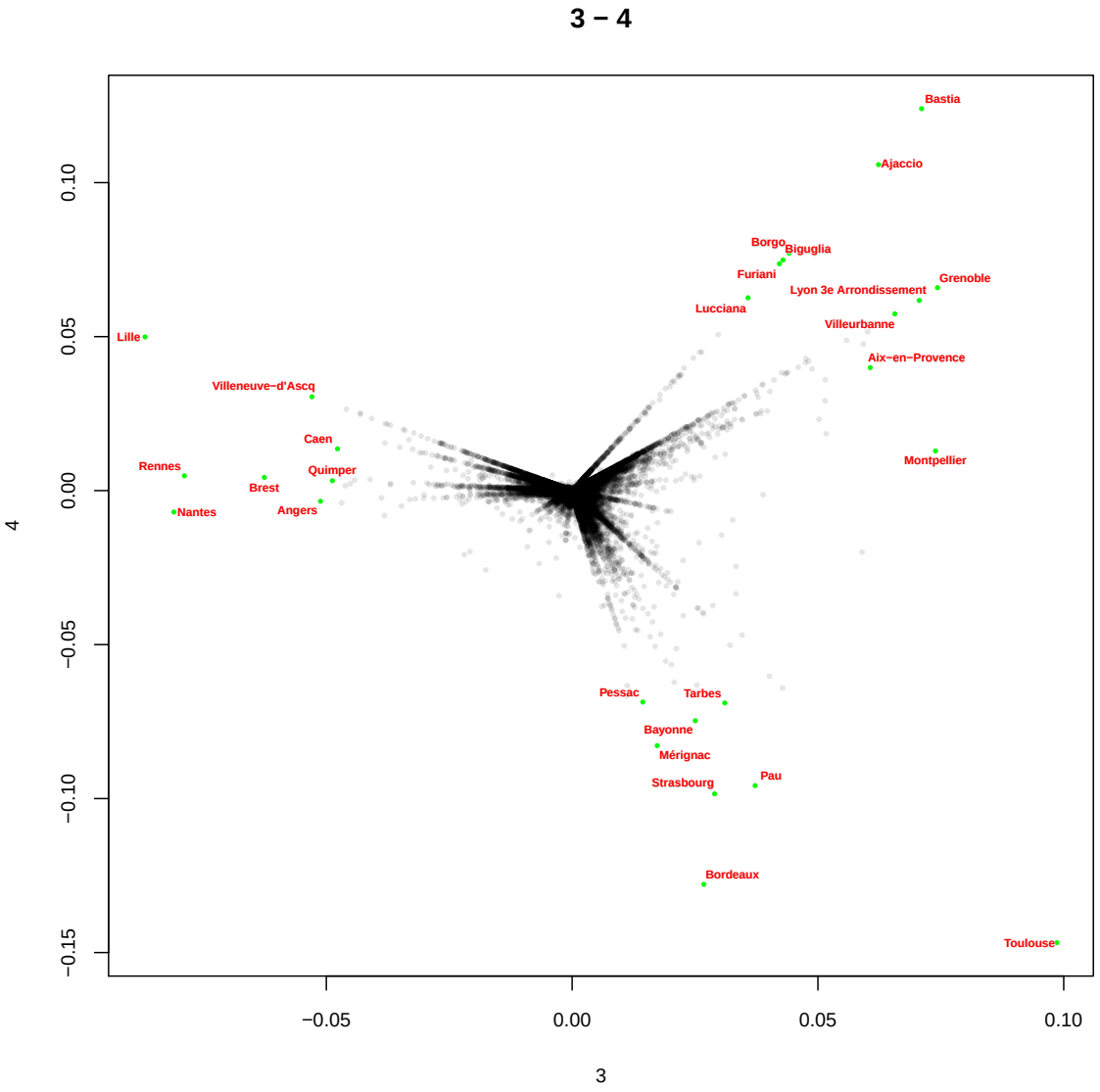
FIGURE 5 – 200 premières valeurs propres de L_{rw}

Premier vecteur propre




1 – 2
Strasbourg
Metz
Mulhouse
Nancy
Colmar
Haguenau
Schiltigheim
Belfort
Lille
Paris 13e Arrondissement
Paris 8e Arrondisseme
Paris 12e Arrondissement
Paris 15e Arrondissement
Nantes
Bayonne
Pessac
Colomiers
Blagnac
Mérignac
Pau
Bordeaux
Toulouse
0.15
0.10
0.05
0.00
−0.05
−0.10
2
0.00
0.02
0.04
0.06
0.08
1


2 − 3
Toulouse
Montpellier
Bastia
Grenoble
Lyon 3e Arrondissement
Ajaccio
Villeurbanne
Aix−en−Provence
Blagnac
Colomiers
Pau
Bordeaux
Bayonne
Mérignac
Pessac
Mulhouse
Colmar
Belfort
Schiltigheim
Haguenau
Metz
Nancy
Strasbourg
Caen
Quimper
Angers
Villeneuve−d'Ascq
Brest
Rennes
Nantes
Lille
0.10
0.05
0.00
−0.05
3
−0.10
−0.05
0.00
0.05
0.10
0.15
2


3 − 4
Bastia
Ajaccio
Borgo
Biguglia
Furiani
Grenoble
Lyon 3e Arrondissement
Lucciana
Villeurbanne
Lille
Aix−en−Provence
Villeneuve−d'Ascq
Caen
Montpellier
Rennes
Quimper
Brest
Angers
Nantes
Pessac
Tarbes
Bayonne
Mérignac
Strasbourg
Pau
Bordeaux
Toulouse
0.10
0.05
0.00
−0.05
−0.10
−0.15
4
−0.05
0.00
0.05
0.10
3

Laplacien régularisé

- comme souvent, la classification spectrale fonctionne bien sur des partitions **clairement séparées**
- ce qui n'est pas forcément le cas en pratique

  ce qui se manifeste par l'hétérogénéité des degrés des sommets

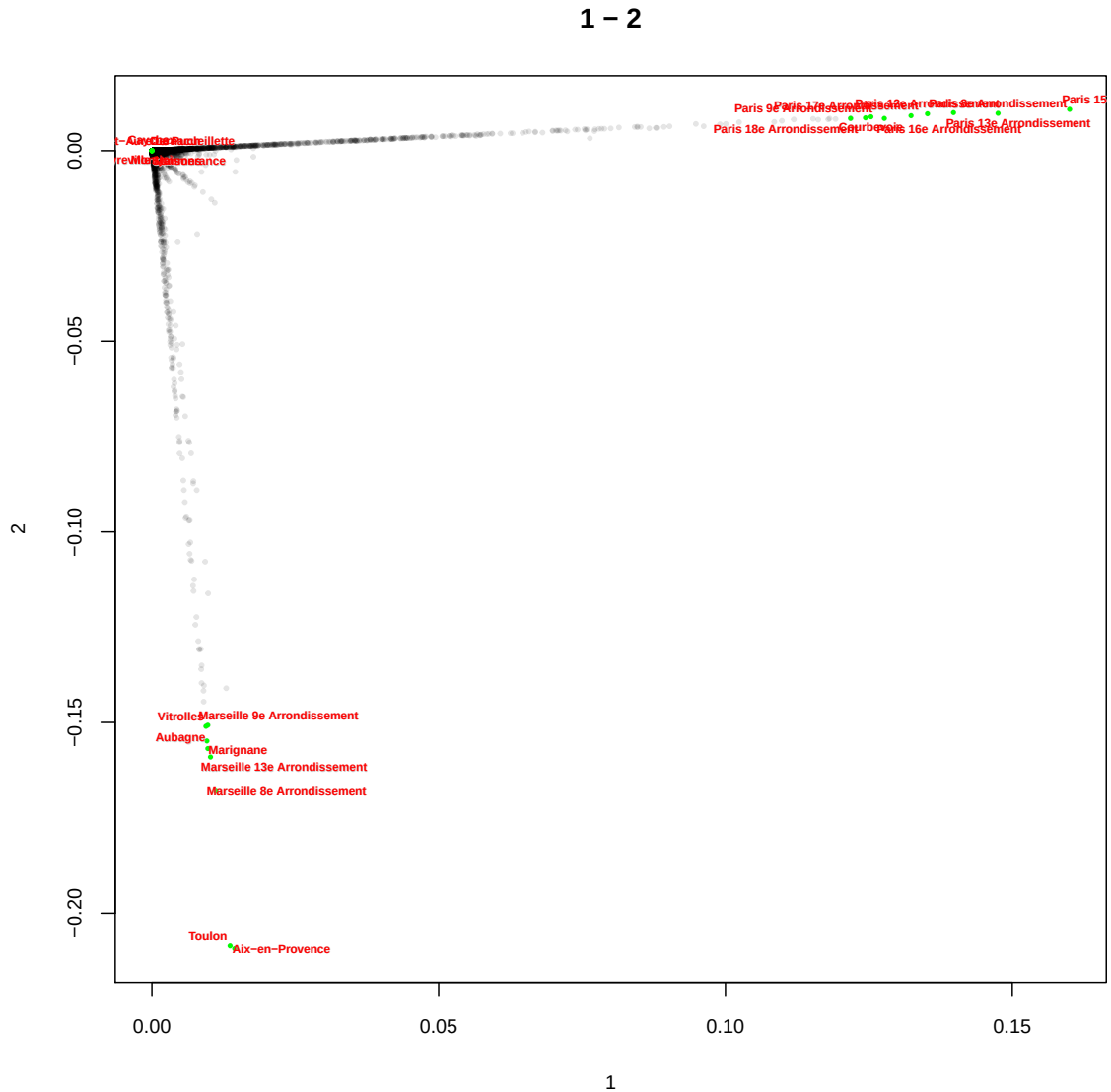
- alternative **pour tempérer** l'effet de l'hétérogénéité, **le laplacien régularisé** :

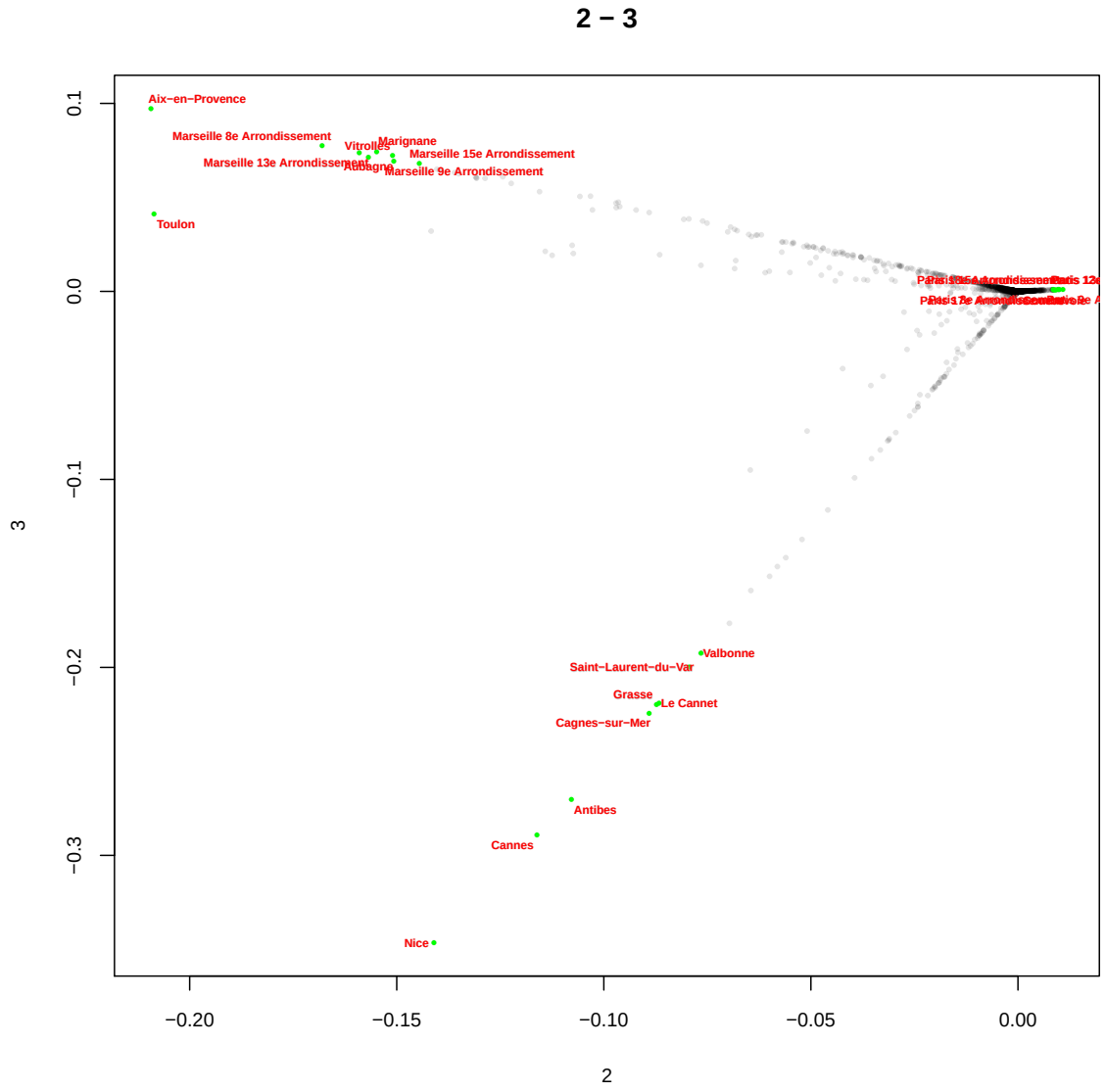
$$\mathbf{L}_\tau = \mathbf{D}_\tau^{-1/2}\mathbf{W}\mathbf{D}_\tau^{1/2} \tag{25}$$

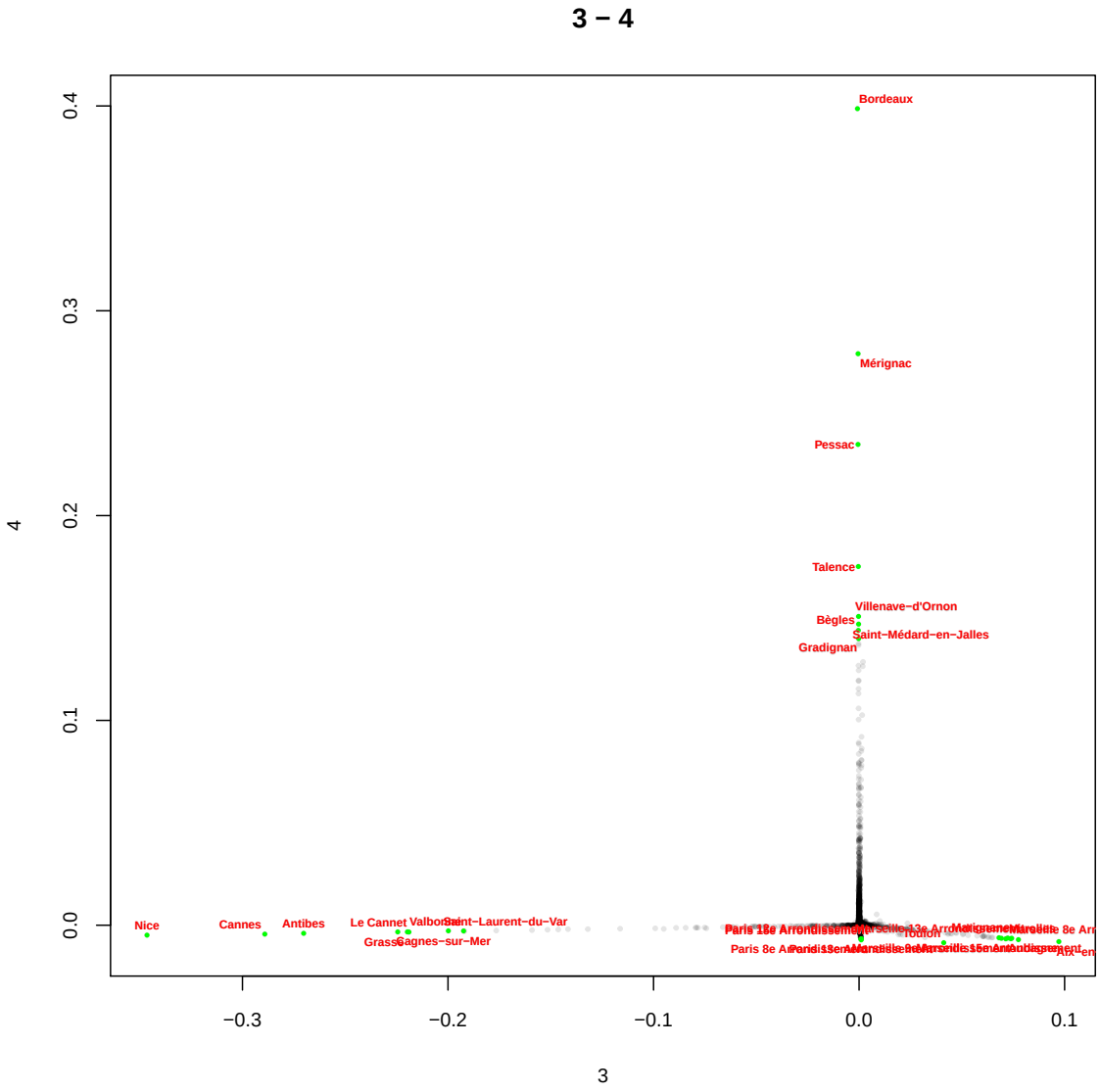
  avec $\mathbf{D}_\tau = \mathbf{D} + \tau\mathbf{I}$

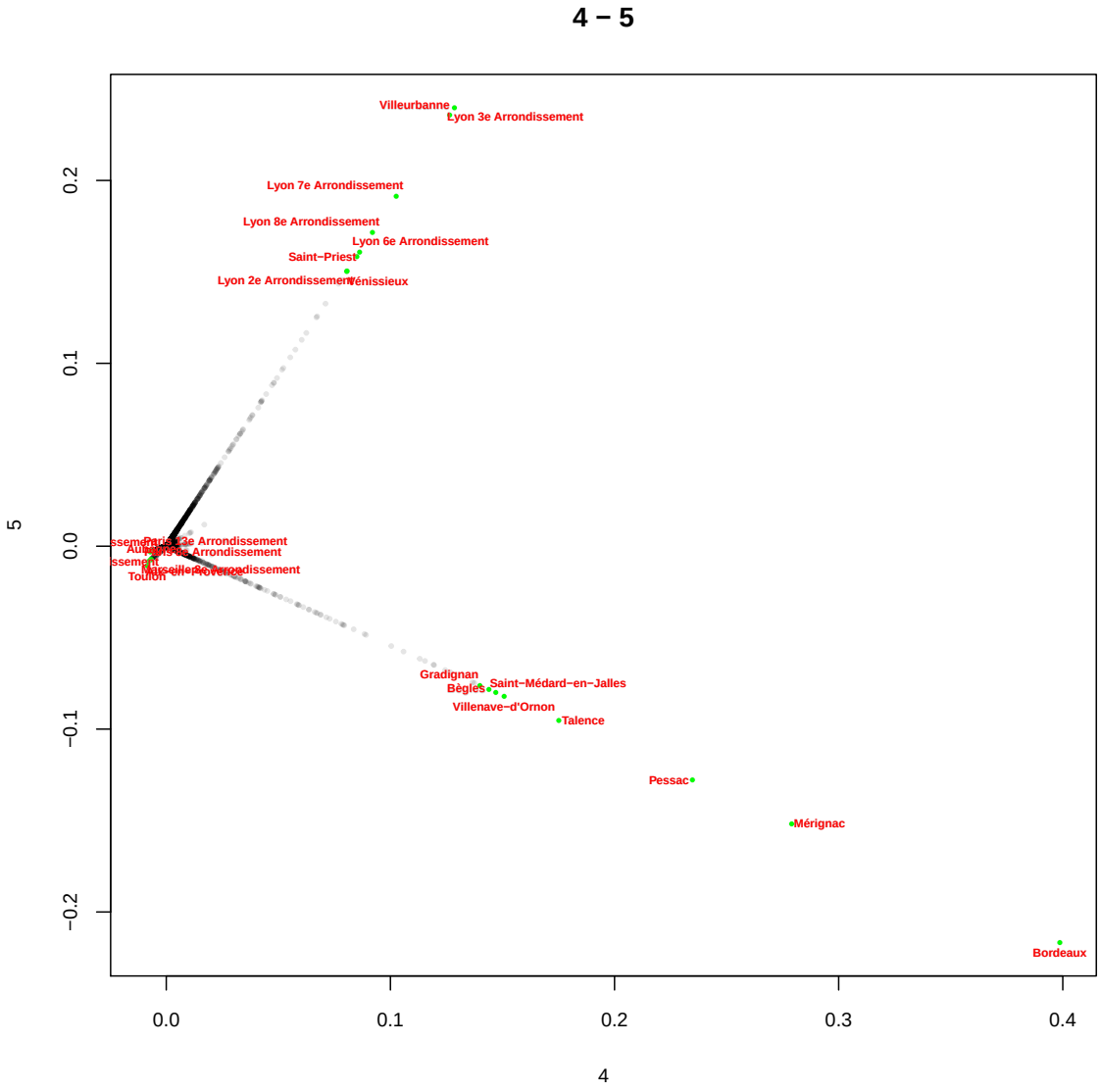
- τ peut prendre **différentes valeurs**

  $\tau = \bar{d}_v$ marche bien en théorie et en pratique


1 − 2
Paris 15
Paris 13e Arrondissement
Paris 18e Arrondissement
Paris 16e Arrondissement
Vitrolles
Marseille 9e Arrondissement
Aubagne
Marignane
Marseille 13e Arrondissement
Marseille 8e Arrondissement
Toulon
Aix−en−Provence
0.00
−0.05
−0.10
−0.15
−0.20
2
0.00
0.05
0.10
0.15
1


2 − 3
Aix−en−Provence
Marseille 8e Arrondissement
Vitrolles
Marignane
Marseille 13e Arrondissement
Marseille 15e Arrondissement
Aubagne
Marseille 9e Arrondissement
Toulon
Valbonne
Saint−Laurent−du−Var
Grasse
Le Cannet
Cagnes−sur−Mer
Antibes
Cannes
Nice
0.1
0.0
−0.1
−0.2
−0.3
3
−0.20
−0.15
−0.10
−0.05
0.00
2


3 − 4
Bordeaux
Mérignac
Pessac
Talence
Villenave−d'Ornon
Bègles
Saint−Médard−en−Jalles
Gradignan
Nice
Cannes
Antibes
Le Cannet
Valbonne
Saint−Laurent−du−Var
Grasse
Gagnes−sur−Mer
Toulon
0.4
0.3
0.2
0.1
0.0
4
−0.3
−0.2
−0.1
0.0
0.1
3


4 – 5
Villeurbanne
Lyon 3e Arrondissement
Lyon 7e Arrondissement
Lyon 8e Arrondissement
Lyon 6e Arrondissement
Saint−Priest
Lyon 2e Arrondissement
Vénissieux
Gradignan
Bègles
Saint−Médard−en−Jalles
Villenave−d'Ornon
Talence
Pessac
Mérignac
Bordeaux
Toulon
0.2
0.1
0.0
−0.1
−0.2
5
0.0
0.1
0.2
0.3
0.4
4

Conclusion

Conclusion

- **l'analyse spectrale** du laplacien du graphe est une méthode performante pour dégager **des sous-ensembles cohésifs**
- mais elle est notamment sensible à **l'hétérogénéité de la distribution** des degrés des sommets
- la régularisation permet toutefois **d'en tempérer les effets**
- plus généralement, l'algèbre linéaire offre **de nombreuses possibilités** pour :
  - le stockage
  - la manipulation
  - l'analyse
- **de grands graphes**

Merci pour votre attention

Bibliographie

Bibliographie I

Contamin, Jean-Gabriel, Thomas Léonard et Thomas Soubiran (2017), « Les transformations des comportements politiques au prisme de l'e-pétitionnement. Potentialités et limites d'un dispositif d'étude pluridisciplinaire ». *Réseaux*, 204, 4, p. 97–131.

Kepner, Jeremy et John Gilbert, éds. (2011), *Graph Algorithms in the Language of Linear Algebra*. SIAM. 375 p.

Luxburg, Ulrike von (2007), « A tutorial on spectral clustering ». *Statistics and Computing*, 17, p. 395–416.

Qin, Tai et Karl Rohe (2013), « Regularized Spectral Clustering under the Degree-Corrected Stochastic Blockmodel ». *arXiv e-prints*. arXiv :1309.4111.