Web Mining – Détection des communautés

Nous travaillons sous Python dans cet exercice, en particulier avec le package « igraph ».

A. Documentation

- Notre support privilégié est : « Détection de communautés – Diapos », http://tutoriels-data-science.blogspot.com/p/tutoriels-en-francais.html#7725326059341656867, les numéros de page dans le texte y feront référence.
- La documentation du package « igraph » est en ligne (https://igraph.org/python/doc/api/ ; https://python.igraph.org/en/stable/).
- [**Vidéo 1**] https://www.youtube.com/watch?v=Vp5-d-lk9jA (Clustering – Détection de communautés)
- [**Vidéo 2**] https://www.youtube.com/watch?v=VjVr0-dCJlA (Multidimensional Scaling sous Python)

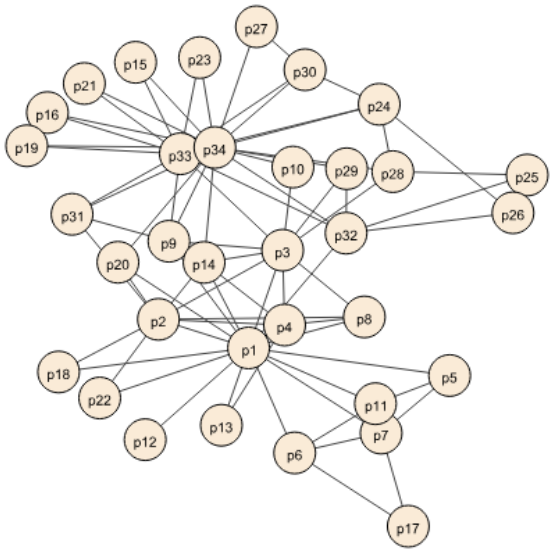
B. Création de l'environnement

Il nous faut au moins les packages suivants : pandas, scikit-learn, seaborn, pycairo, **igraph**. Pour ceux qui en ont l'usage, j'ai mis le fichier « **envtd_igraph.yml** » sur notre drive.

C. Données « Karate »

Le fichier « **karate.txt** » décrit une matrice d'adjacence. Il provient d'un problème notoire d'analyse des réseaux sociaux : le « **Club de Karaté de Zachary** ». Les membres du club, à force de se tirer dans les pattes (mais pas sur le tatami) ont fini par provoquer une scission. Les deux leaders, le président et l'instructeur, ont chacun créé une nouvelle structure. Chaque membre a dû choisir son camp.

1. Visualisez les données dans un éditeur de texte. Que remarquez-vous concernant les étiquettes de lignes et de colonnes ? (la première ligne et la première colonne correspondent à des identifiants)
2. Chargez le fichier en respectant sa structure. Quelle est la dimension du data.frame ? (34, 34)
3. Affichez la liste des personnages. (p1, p2, ..., p34)
4. Comptabilisez le nombre d'arêtes dans le graphe (sans prendre en compte les doublons) ? (78)
5. Importez la librairie « igraph ». Affichez le numéro de version (0.11.4 pour moi) (**Vidéo 1**, 15:28)
6. Transformez la matrice d'adjacence en une structure de graphe (Graph.adjacency). Attention, notre graphe n'est pas orienté. Affichez le type de l'objet instancié (igraph.Graph)
7. Affichez la structure obtenue, combien de sommets disposons-nous ? (34) De connexions ? (78).
8. Spécifiez les « labels » (étiquettes) des sommets, puis affichez le graphe (.plot)



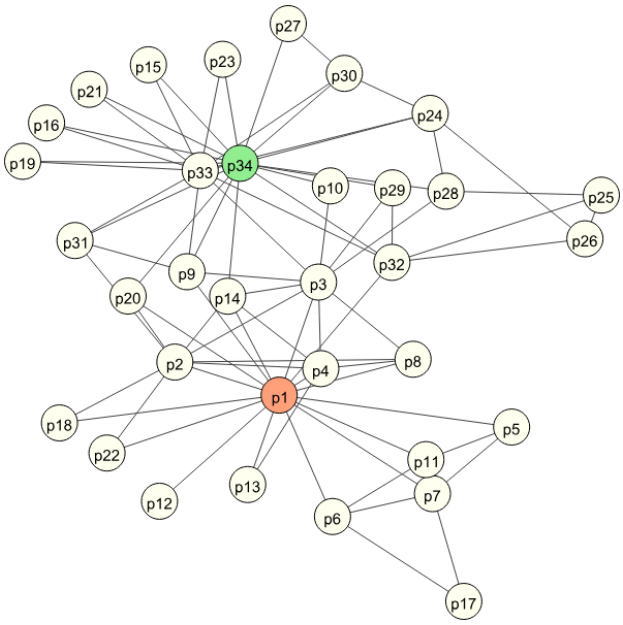
9. Quel est le nombre de voisins du sommet « p11 » ? (3) Effectuez le calcul à l'aide de la matrice initiale, puis à l'aide de la fonction dédiée du graphe (.neighborhood_size) (cf. https://igraph.org/python/doc/api/igraph.Graph.html pour les propriétés et méthodes de **Graph**).
10. Ah ? Il y a une incohérence apparemment ? Affichez la liste des voisins concernés (.neighborhood). Que constatez-vous ? ([10, 0, 4, 5])
11. Disposer des indices c'est bien, afficher les étiquettes c'est mieux. (['p11', 'p1', 'p5', 'p6']) (cf. du côté de la propriété « vs » qui a permis d'intégrer les étiquettes dans la structure du graphe).

D. Centralité

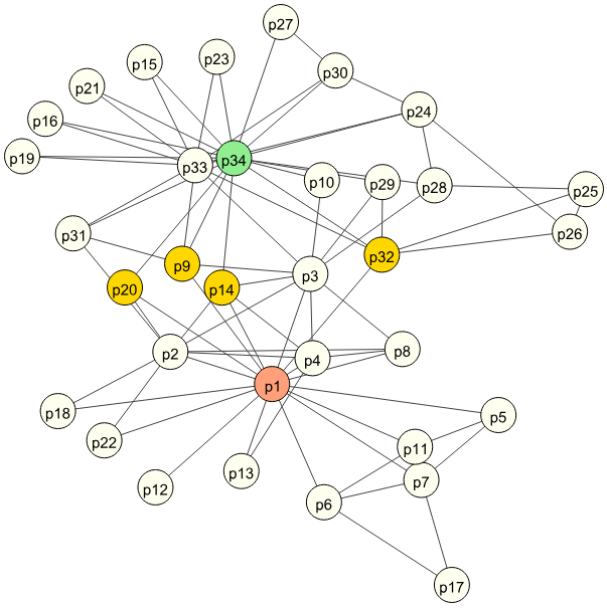
12. En utilisant la formule vue en cours, calculez le degré de centralité absolu de chaque sommet à partir de la matrice d'adjacence (page 9). Quels sont les 3 personnages les plus centraux ? (p34 : 17, p1 : 16, p33 : 12).
13. Qu'obtenez-vous avec la fonction proposée par le package ? (.degree).
14. Calculez et affichez la matrice des distances géodésiques (.distances). Attention, quel type de structure obtenons-nous ?
15. Quelle est sa longueur ? (len) Et celle du premier élément ? Que peut-on en déduire ?
16. Transformez la structure en matrice « numpy ». Vérifiez ses dimensions.
17. Quelle est la valeur maximale de la distance ? (5).
18. Affichez la liste des paires de sommets concernés c.-à-d. qui sont le plus éloignés les uns des autres (affichez quelque chose qui ressemblerait à ceci : [('p15', 'p17'), ('p16', 'p17'), ('p17', 'p19'), ('p17', 'p21'), ('p17', 'p23'), ('p17', 'p24'), ('p17', 'p27'), ('p17', 'p30')]). Attention, le graphe est non-orienté, vous ne devez pas affichez deux fois la même connexion.
19. Calculez la centralité moyenne des sommets (page 9). Quelles sont les 3 sommets avec les plus faibles valeurs de cette centralité ? (p1 : 1.757, p3 : 1.787, p34 : 1.818). Affichez l'inverse (1/x) de ces valeurs (p1 : 0.568, p3 : 0.559, p34 : 0.55)
20. Calculez la closeness centrality des sommets avec l'outil dédié (.closeness). Quels sont les 3

sommets avec les plus fortes valeurs ? (p1 : 0.568, p3 : 0.559, p34 : 0.55). Les résultats sont cohérents avec ceux de la question précédente ? (oui, heureusement)

21. Calculez la betweenness centrality des sommets (.betweenness). Quels sont les 3 sommets avec les plus fortes valeurs ? (p1 : 231.0714286 ; p34 : 160.5515873 ; p33 : 76.6904762)
22. Que pensez-vous de tout cela ? Affichez de nouveau le graphe en mettant en évidence les deux éléments à forte centralité que semblent être **p1** et **p34** (pour attribuer une couleur spécifique à un sommet identifié par son numéro : `g.vs[numéro]['color'] = code couleur` ; **Vidéo 1**, 29:00)

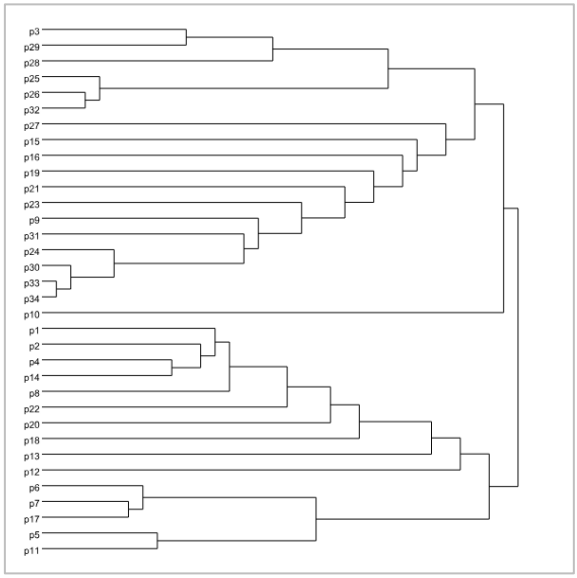


23. On s'intéresse aux individus relais qui pourraient faire la jonction entre « p1 » et « p34 ». Quels sont les individus situés sur le plus court chemin reliant « p1 » à « p34 » ? (.get_all_shortest_paths) (31, 19, 13, 8). Mettez-les en évidence dans le graphe.

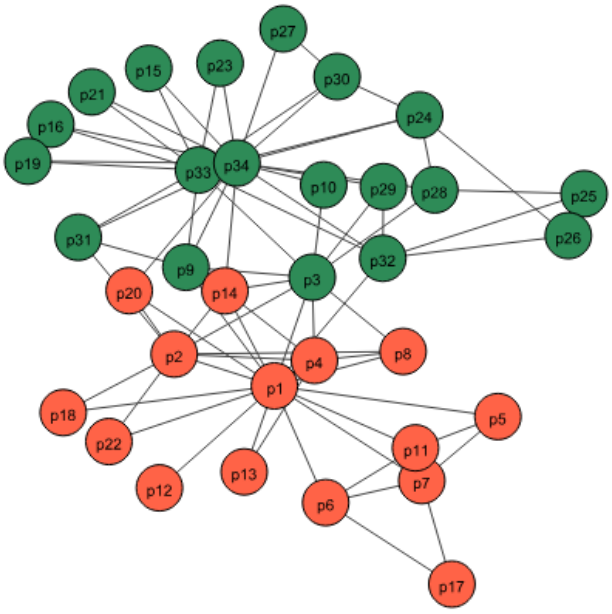


E. Détection des communautés – Edge betweenness

24. Nous décidons d'identifier les communautés à l'aide de l'approche divisive (edge betweenness) (page 19) (https://igraph.org/python/doc/api/igraph.Graph.html#community_edge_betweenness) (.community_edge_betweenness). Affichez le dendrogramme (igraph.plot)



25. Via l'objet obtenu (https://igraph.org/python/doc/api/igraph.clustering.VertexDendrogram.html), spécifiez un découpage en 2 classes (.as_clustering). Affichez la liste des sommets par cluster (print).
26. Recensez les groupes d'appartenance de chaque sommet (.membership). Quel est l'effectif de chaque groupe ? (.sizes) (15, 19)
27. Affichez le graphe en mettant des couleurs distinctives pour chaque groupe. Comment situez-vous les deux sommets qui se sont distingués précédemment ?



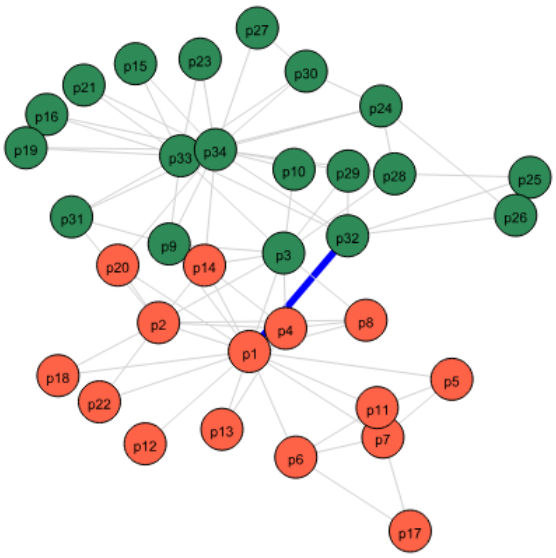
28. La mesure « edge betweenness » représente la fréquence avec laquelle une connexion est

empruntée lorsque l'on considère les plus courts chemins entre chaque paire de nœud. Plus la valeur est élevée, plus la connexion est importante parce qu'elle permet de faire la jonction entre les sommets, les groupes de sommets. Calculez sa valeur pour l'ensemble des connexions (https://igraph.org/python/doc/api/igraph._igraph.GraphBase.html#edge_betweenness).

29. Transformez le résultat en vecteur « numpy ». Combien de valeurs disposons-nous ? (78) Est-ce cohérent avec la structure du graphe ? (oui, plutôt, il y a 78 connexions dans le graphe).
30. Affichez les paires de sommets concernés, en leur associant leur distance « edge betweenness » (je suis revenu dans la matrice d'adjacence initiale pour repérer les connexions entre les sommets)

```
p1 - p32     71.392857
p1 - p7      43.833333
p1 - p6      43.833333
p1 - p3      43.638889
p1 - p9      41.648413
                ...
```

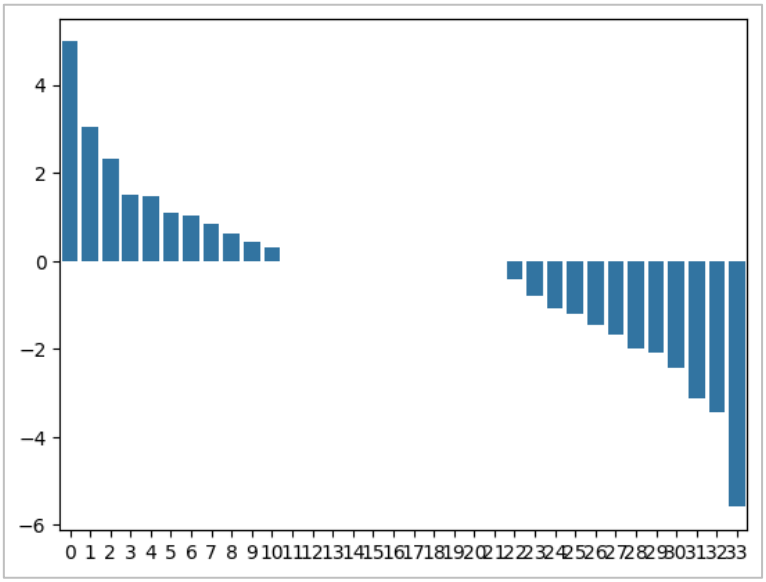
31. Est-ce que les individus concernés par la connexion la plus substantielle permettrait de faire le pont entre les 2 communautés ? (oui, p1 et p32 sont dans des communautés différentes)
32. Mettez la connexion « p1 – p32 » en évidence dans le graphe (cf. la propriété « es » du graphe pour manipuler les connexions, avec `g.es[numéro connexion]['color'] = code couleur` pour indiquer la couleur ; pour l'épaisseur du trait, nous utiliserons « width »).



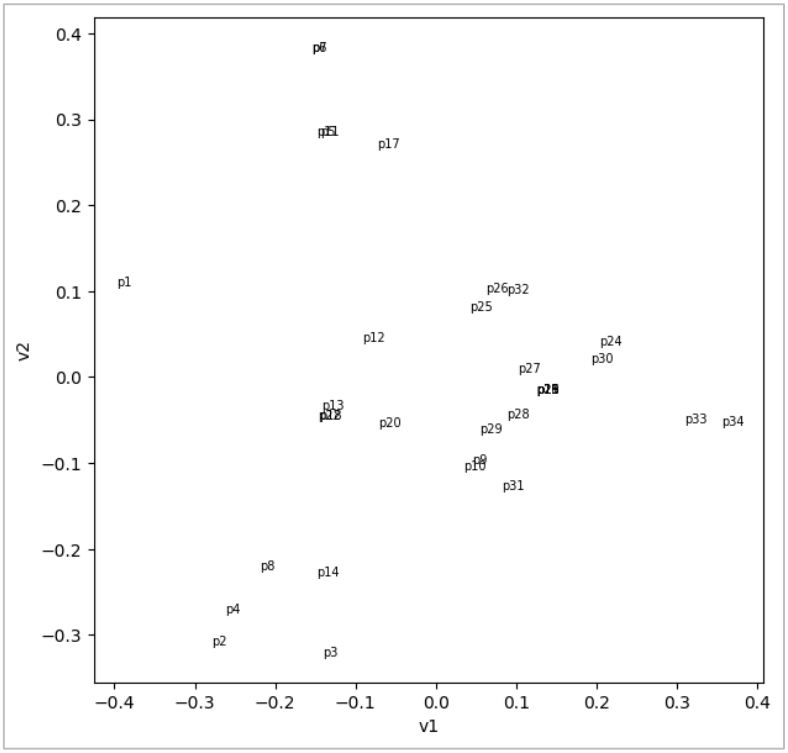
F. Matrice de modularité – Représentation factorielle

33. Calculez la matrice de modularité (.modularity_matrix) (matrice B, page 15). Transformez le résultat en matrice numpy. Vérifiez ses dimensions. (34, 34)
34. Diagonalisez la matrice (https://numpy.org/doc/stable/reference/generated/numpy.linalg.eig.html). Attention : (1) les valeurs propres (et les vecteurs propres associés) ne sont pas triées, nous devons nous en charger nous-même ; (2) les résultats sont exprimés en nombres complexes, sauf

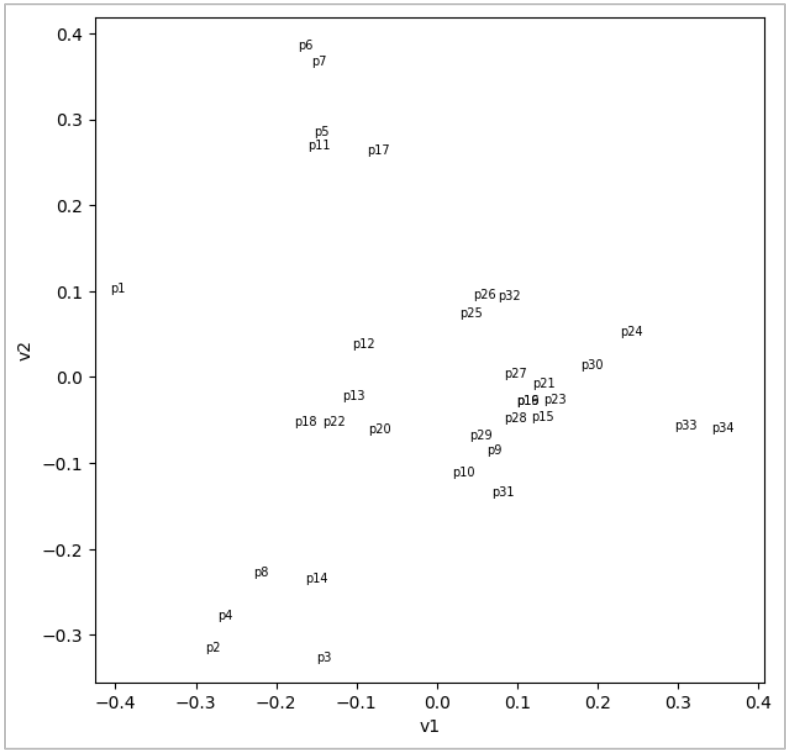
que la partie imaginaire est toujours nulle dans notre cas [la matrice à diagonaliser est symétrique], il nous faut par conséquent tout transformer en réels). Affichez les valeurs propres triées par ordre décroissant à l'aide d'un barplot.



35. Affichez les points dans le premier plan factoriel défini par les vecteurs propres, insérez les étiquettes dans le graphique.



36. Malheureusement, certaines zones du graphique sont peu lisibles parce que des étiquettes se superposent. Essayez d'y remédier en utilisant le package « adjustText » par exemple (https://adjusttext.readthedocs.io/en/latest/). C'est quand-même mieux non ?



37. Reproduisez le même graphique mais en coloriant les individus en fonction des 2 clusters obtenus lors de l'approche divisive basé sur le « edge betweenness » ci-dessus. Les groupes sont-ils bien distincts ? (oui, clairement)

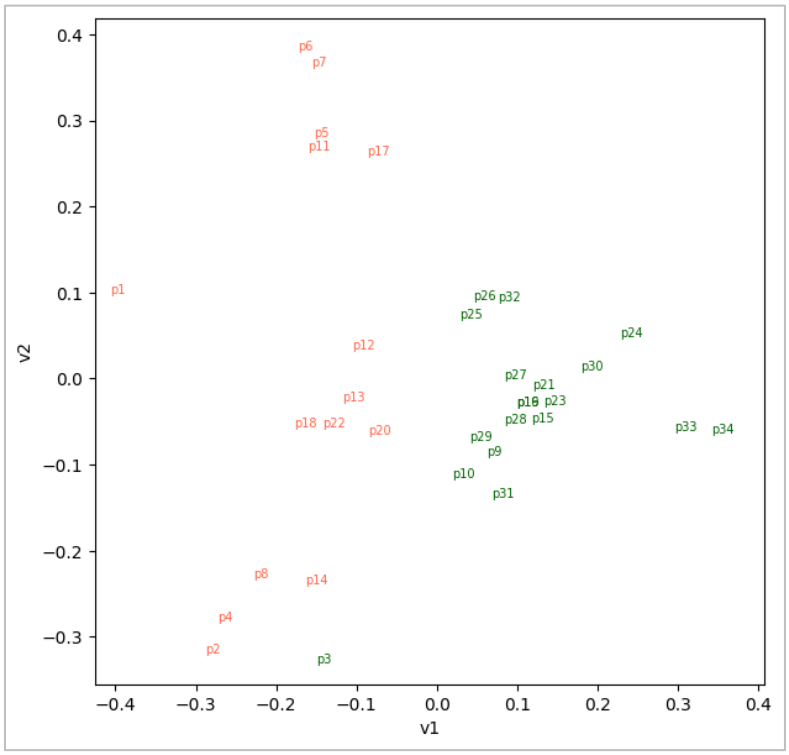


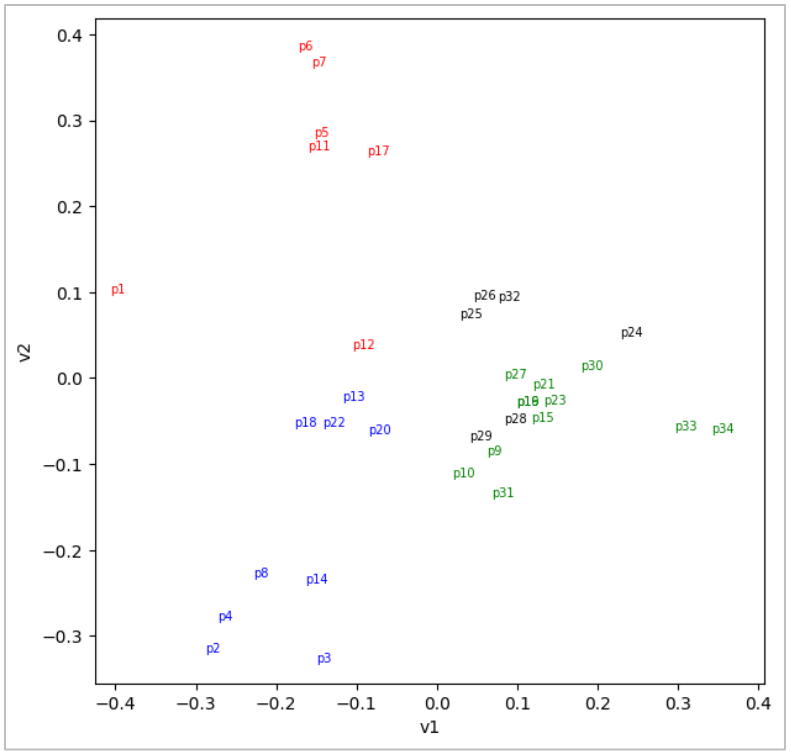
Figure 1 - Position des sommets dans le plan et groupes attribués par l'approche "edge betweenness"

G. Détection par maximisation de la modularité

38. Nous souhaitons maintenant appliquer la méthode basée sur la maximisation de la modularité

(page 22) (https://igraph.org/python/doc/api/igraph._igraph.GraphBase.html#community_leading_eigenvector).

39. Combien de groupes sont proposés ? (4) Quel est l'effectif pour chaque groupe ? (7, 19, 9, 6)
40. Matérialisez-les dans le plan factoriel défini ci-dessus.

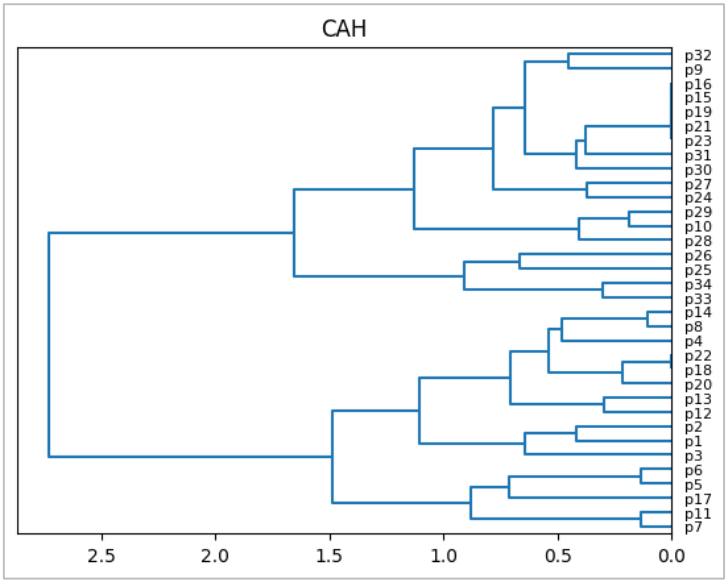


41. Croisez ces groupes avec ceux obtenus à l'aide de l'approche divisive. Observe-t-on une certaine cohérence dans les résultats ?

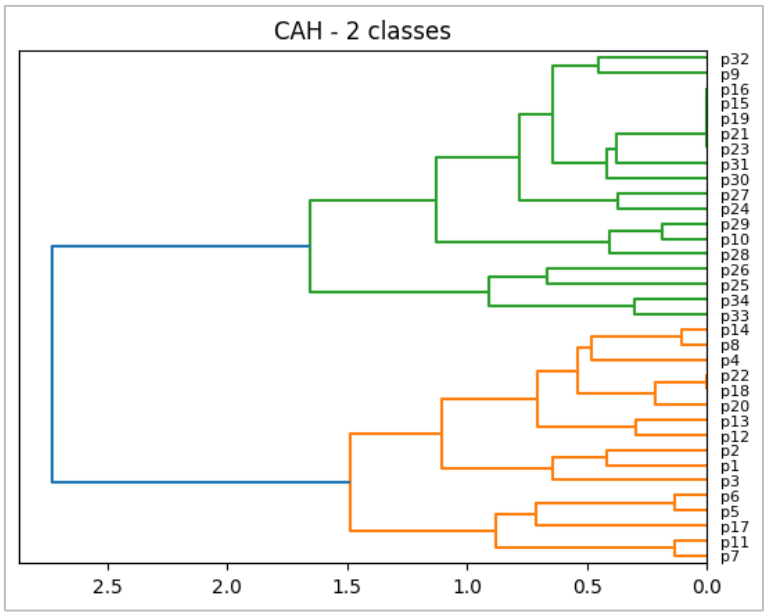
col_0	0	1	2	3
row_0				
0	7	0	8	0
1	0	12	1	6

H. Approche agglomérative

42. On souhaite appliquer la méthode agglomérative (page 16). Calculez la matrice des distances cosinus par paires d'individus (cosine_distances) (**Vidéo 1**, 19:15).
43. Vérifiez qu'elle vérifie les propriétés d'une distance (is_valid_dm) (oui, heureusement)
44. Vectorisez la matrice afin de pouvoir la présenter à la CAH de « scipy » (squareform, **Vidéo 1**, 20:33).
45. Lancez la CAH (classification ascendante hiérarchique) avec la méthode de Ward (ward + dendrogram ; **Vidéo 1**, 20:59).



46. Pour le coup, la partition en 2 classes semble évidente. Reproduisez le dendrogramme en les matérialisant.



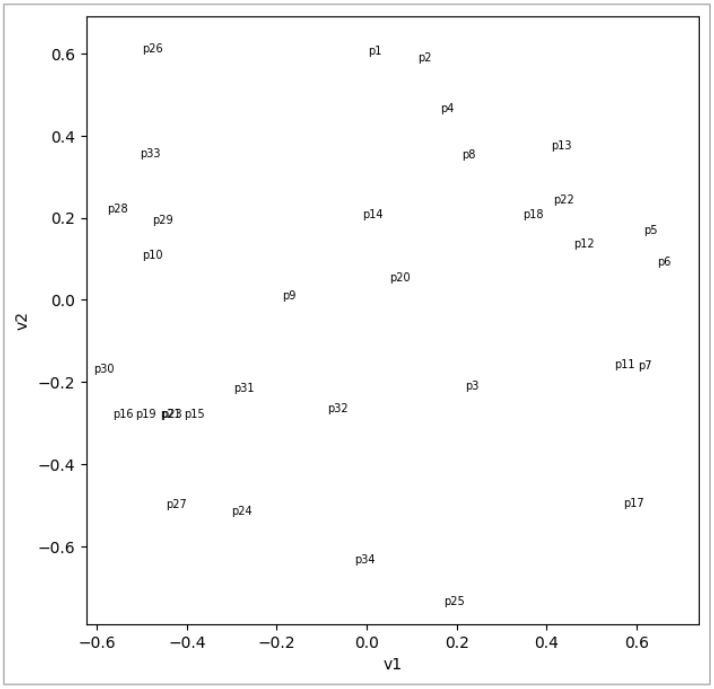
47. Effectuez le découpage en 2 groupes (fcluster) (**Vidéo 1**, 22:30). Est-ce qu'ils sont cohérents avec ceux obtenus avec la première méthode basée sur le « edge betweenness » ? (oui, à 1 individu près)

col_0	1	2
row_0		
0	15	0
1	1	18

48. Qui est cet individu qui a changé de groupe ? (p3) Est-ce qu'on peut le comprendre au regard de sa position dans le plan factoriel et le groupe auquel l'a rattaché l'approche « edge beetweenness » ? (cf. **Figure 1**).

I. Positionnement multidimensionnel + K-Means

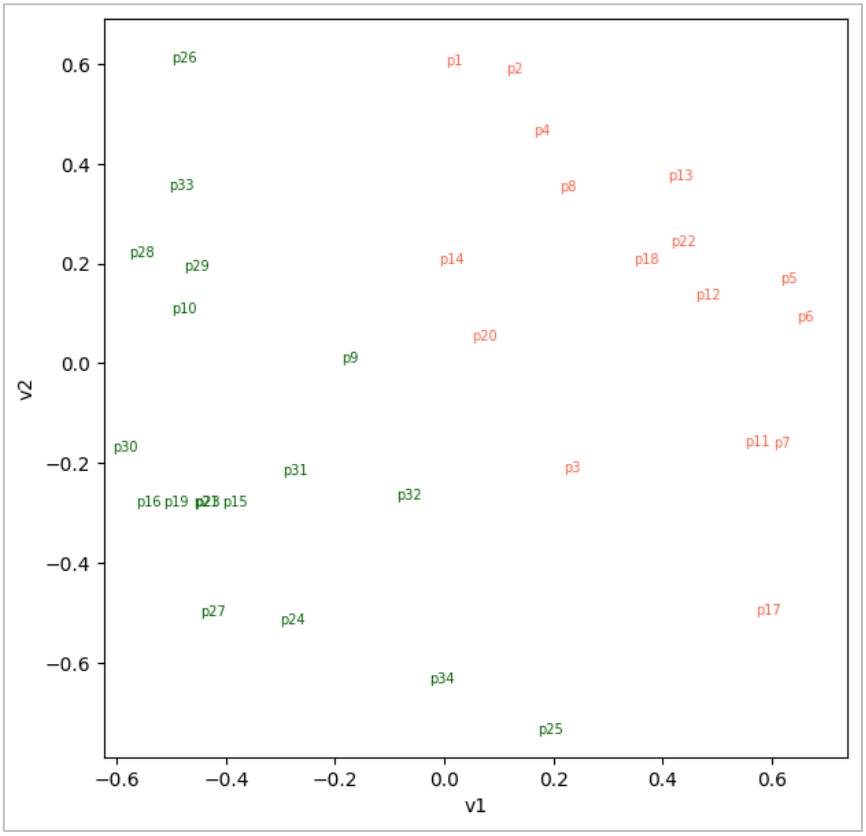
49. Essayons de passer par un positionnement multidimensionnel basé sur la distance cosinus pour réaliser la partition. L'idée globale est décrite dans la **Vidéo 2**, mais nous utiliserons la classe MDS de « scikit-learn » https://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html. Instanciez l'objet pour (n_components = 2). Faites attention à l'option « dissimilarity », nos données se présentent sous la forme d'une matrice de distances déjà précalculée. Nous fixons (random_state = 0) pour obtenir tous les mêmes résultats (modulo la version de scikit-learn).
50. Affichez les coordonnées des sommets dans l'espace factoriel (.embedding_). Vérifiez les dimensions de la matrice (34, 2 ; forcément).
51. Affichez les points dans le repère « factoriel ». Est-ce qu'on devine déjà les groupes dans le plan ? (oui, sans être prix nobel, on devine à peu près ce que l'on pourrait obtenir avec un K-Means dans ce repère)



52. Appliquez la méthode des K-MEANS sur ce plan factoriel en spécifiant le nombre de groupes à (n_clusters = 2) (**Vidéo 2**, 32:39)
53. Croisez les groupes obtenus avec ceux produits par « edge betweenness ». Les résultats sont cohérents (n'oublions pas que les numéros de groupe sont attribués arbitrairement) ? Qui est l'individu fauteur de trouble ? (p3, c'est le chat noir)

col_0	0	1
row_0		
0	0	15
1	18	1

54. Illustrez les sommets selon leur groupe d'appartenance dans le plan factoriel.



55. Si le cœur vous en dit, et si vous n'êtes pas épuisé, essayez de reconstituez le graphe (sommets + connexions) dans le plan factoriel, avec toujours une matérialisation des groupes d'appartenance (**Vidéo 2**, 34:30).