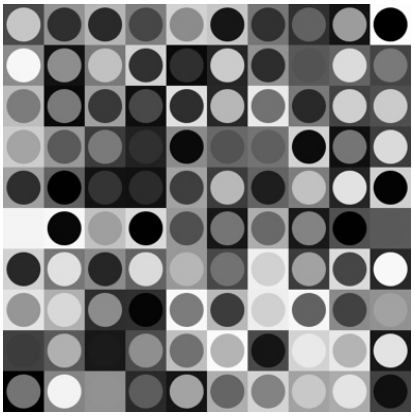
TP3 Visualisation d'informations : dessins de graphes avec Processing
P. Cubaud <cubaud @ cnam.fr>

L'objectif de ce TP est de découvrir l'environnement Processing et de l'utiliser en mode "immédiat" pour des dessins de graphes de grandes dimensions. Nous utiliserons les fonctions interactives de Processing à la séance suivante.

Pour installer Processing sur votre machine, aller sur le site :
https://processing.org/download/

Processing peut s'utiliser en mode Java, Python ou JavaScript. Il existe aussi un mode R encore expérimental. Utilisez le langage que vous préférez (mais les corrigés sont en Python). Il faut pour cela aller dans le bouton en haut à droite ("Java") et cliquer sur "ajouter un mode".

Exercice 1. Fonctions de dessin de base



Les fonctions de base du dessin avec processing sont :
size() background() fill() stroke() rect() ellipse()

Lire la documentation de ces fonctions (accessible depuis le menu "help")

Ecrire ensuite un programme qui dessine un pavage de carrés et de cercles remplis d'un niveau de gris tiré au hasard, comme ci-contre.

Exercice 2. Parallel coordinate plot

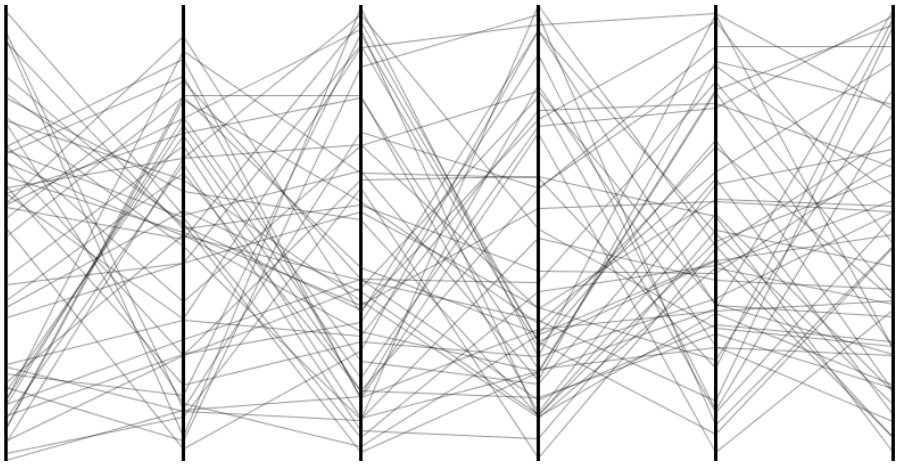
Le "parallel coordinate plot" (P-plot) est une technique de représentation de données multidimensionnelles, promue par A. Inselberg en 1959 et objet depuis de nombreux perfectionnements.

Ecrire un code qui dessine un P-plot sur P > 1 axes avec N lignes de données. On supposera que les données sont stockées dans une matrice data[N][P] de type float et qu'elles ont déjà été normalisées dans une autre partie du programme qui ne nous intéresse pas ici : $0 \leq$ data[i][j] ≤ 1 pour tous $0 \leq i < N$ et $0 \leq j < P$.

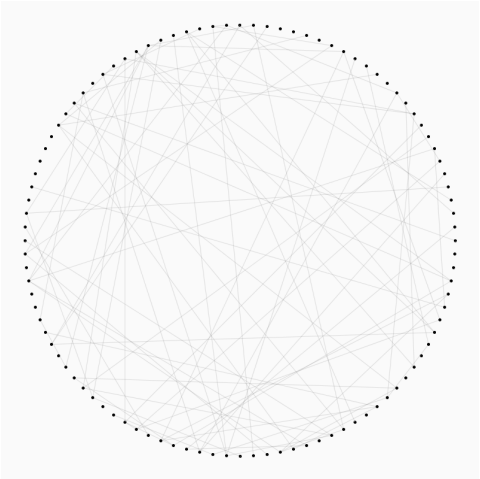
On veut que le code dessine le P-plot en tenant compte de la taille de la fenêtre de l'application (utilisation des variables d'environnement *width* et *height*). Les traits des trajectoires et les P axes seront noir, le fond sera blanc. Dans cette version minimaliste, on ne dessine rien de plus (pas de noms des facteurs sur les axes, de graduations, etc.)

Le résultat du dessin est stocké en fin de programme dans un fichier.

La figure ci-dessous montre un P-plot avec P=6 et N=50. Les données data[i][j] ont été générées par loi uniforme[0,1] .

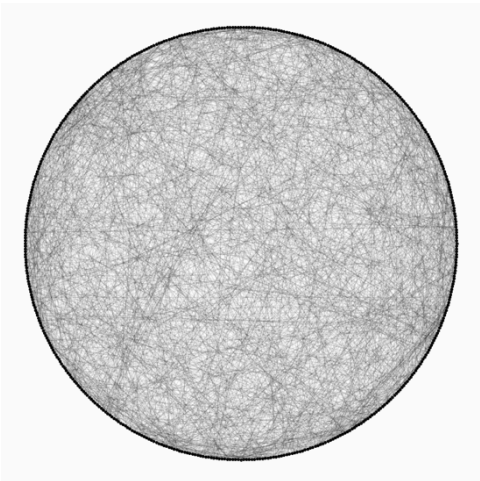


Exercice 3. Graphes circulaires



On souhaite dessiner un graphe avec la répresentation radiale, où tous les sommets sont régulièrement répartis sur un cercle et les arcs qui les relient traversent le cercle (voir les figures ci-contre).

La topologie du graphe sera définie par sa matrice de précédence notée pred[i][j], avec 0≤i<N et 0≤j<N, N étant le nombre de sommets et fixé à l'avance. On prendra comme convention que pred[i][j] vaut 1 s'il existe un arc reliant i à j, sinon pred[i][j] est nul.



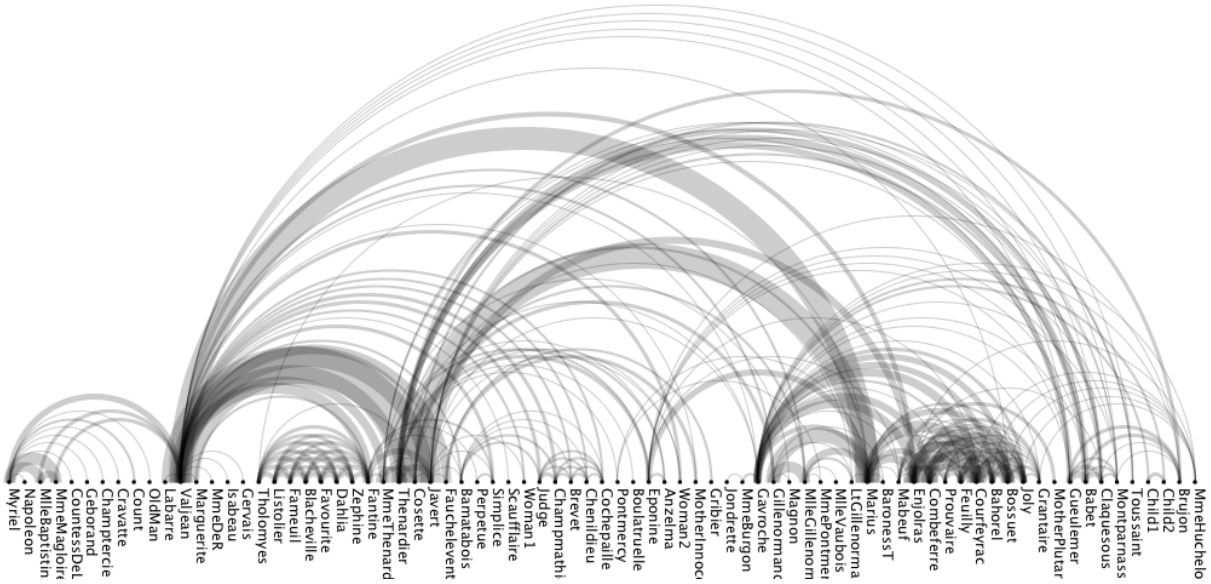
Comme dans l'exercice précédent, on ne cherche pas à travailler sur des "vraies" données, pour se focaliser sur l'impact visuel des dimensions du graphe (nombre de sommets, nombre d'arcs). Ici, pour chaque case pred[i][j] de la matrice, on tirera au sort sa valeur selon une loi de Bernouilli de probabilité P fixée à l'avance.

On veut que le code dessine l'arbre en tenant compte de la taille de la fenêtre de l'application (utilisation des variables d'environnement *width* et *height*). Les traits des arcs seront noirs avec un peu de transparence, le fond sera blanc.

Le résultat du dessin sera stocké en fin de programme dans un fichier.

Les figures ci-contre montrent des graphes avec respectivement (N=100, P=0.08) et (N=500, P=0.08).

BONUS Exercice 4 : Graphe linéaire avec données réelles.



On se propose de visualiser le graphe des interactions entre les personnages du fameux roman de Victor Hugo : Les Misérables. A partir des fichiers fournis (arcs.csv et sommets.csv), écrire un programme Processing qui produit l'image ci-dessus.

Le fichier sommets.csv contient le nom de chaque personnage et un identifiant numérique pour chacun.

Chaque ligne du fichier arcs.csv décrit les échanges entre deux personnages, dont on donne les identifiants dans la colonne "Source" et la colonne "Target". La colonne "Weight" donne le nombre total des échanges entre les deux personnages. Les arcs circulaires ont une épaiseur (strikeWeight) proportionnelle à ce nombre. On utilisera de la transparence dans le tracé des arcs pour rendre le graphe plus lisible.

La lecture de fichiers CSV est facile avec la classe Table et ses méthodes. Voir la documentation sur https://py.processing.org/reference/loadTable.html