Matrix Data Visualization

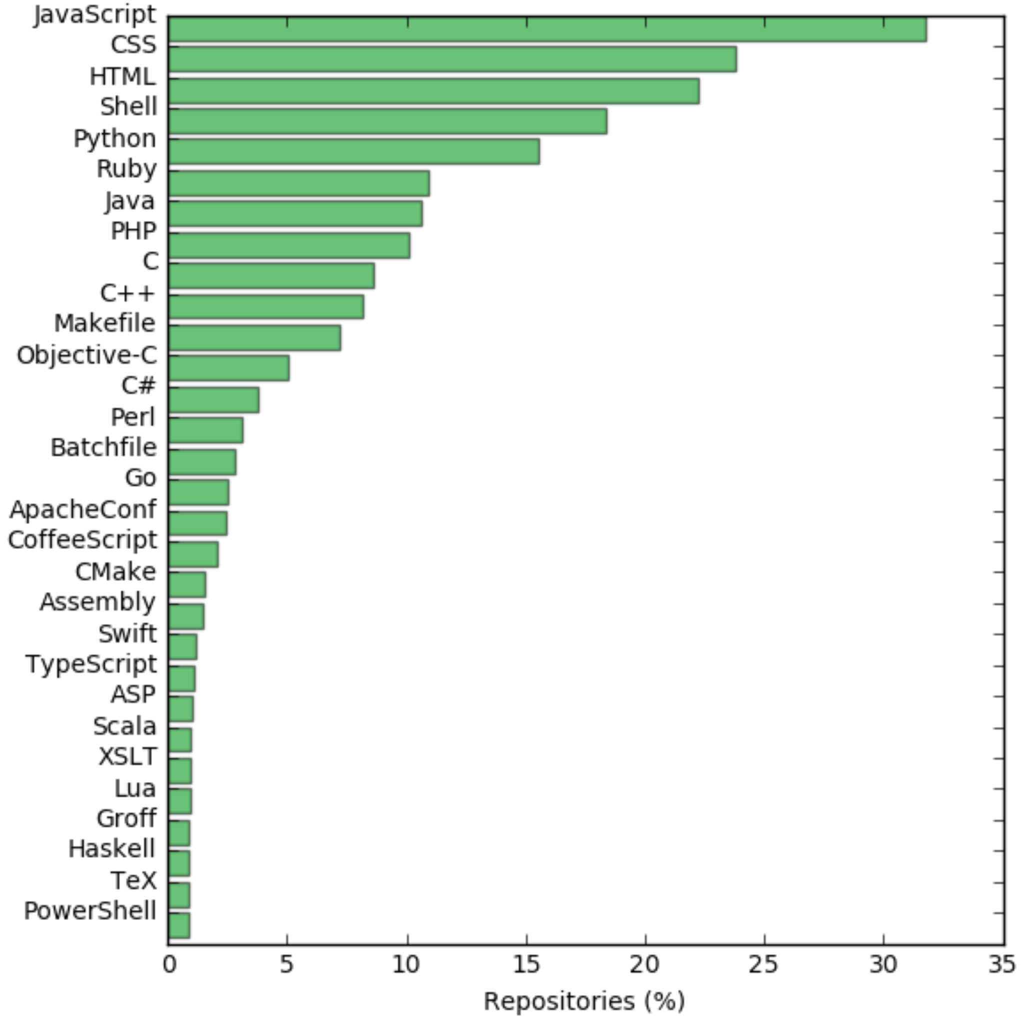
Milan Vojnovic

ST445 Managing and Visualizing Data

GitHub archive dataset

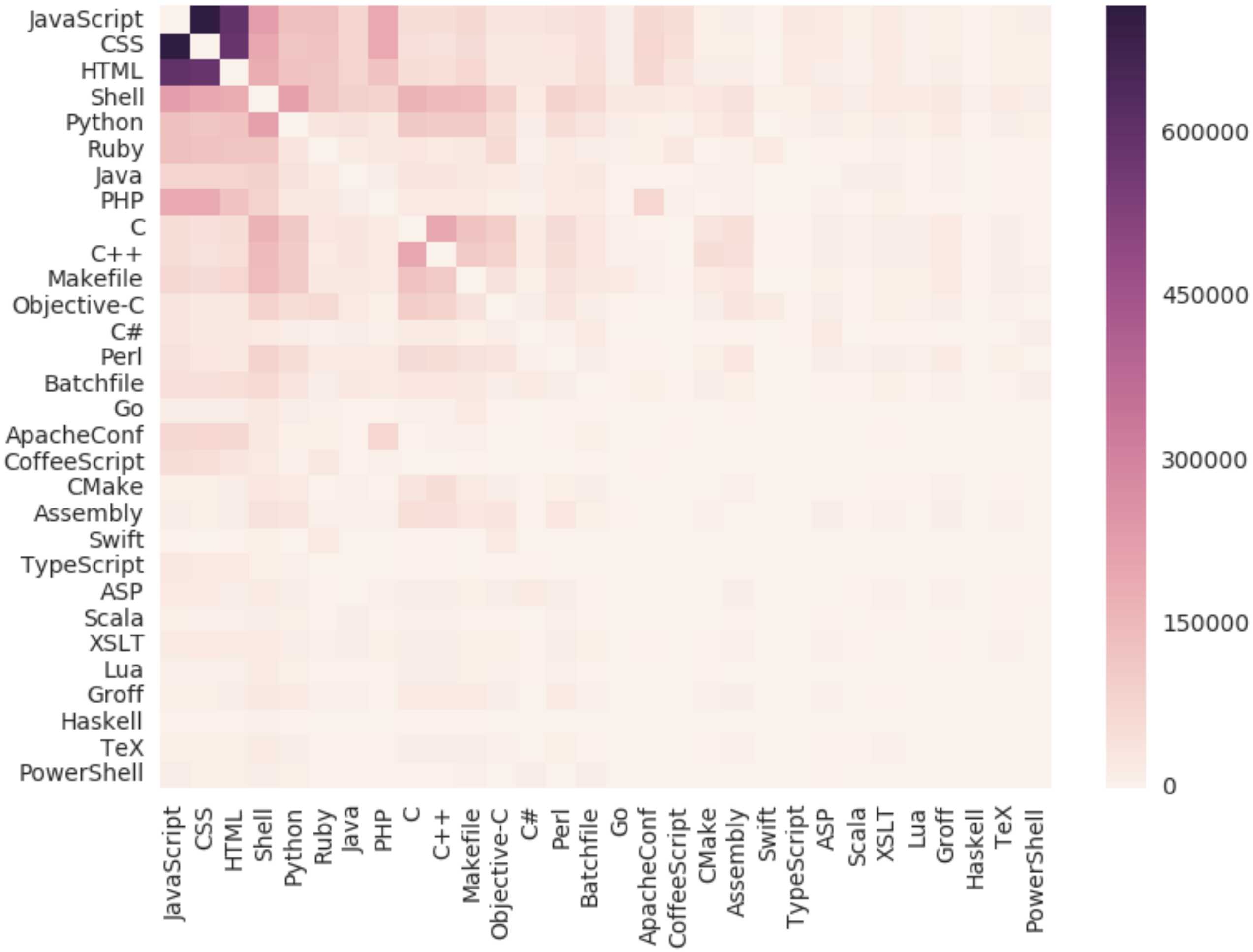
- The plot on this slide shows the percentage of repositories that use specific programming languages

- Suppose our goal is to visualize co-occurrence of pairs of programming languages in different repositories



Heatmap

- Co-occurrence of pairs of programming languages in repositories can be visualized by a heatmap
- It is important how the rows and columns are sorted to visualize any possibly existing clusters
- In this slide, the rows and columns are sorted in decreasing popularity of programming languages
  - A heuristic that for this instance reveals some clusters, but this is not necessary in general

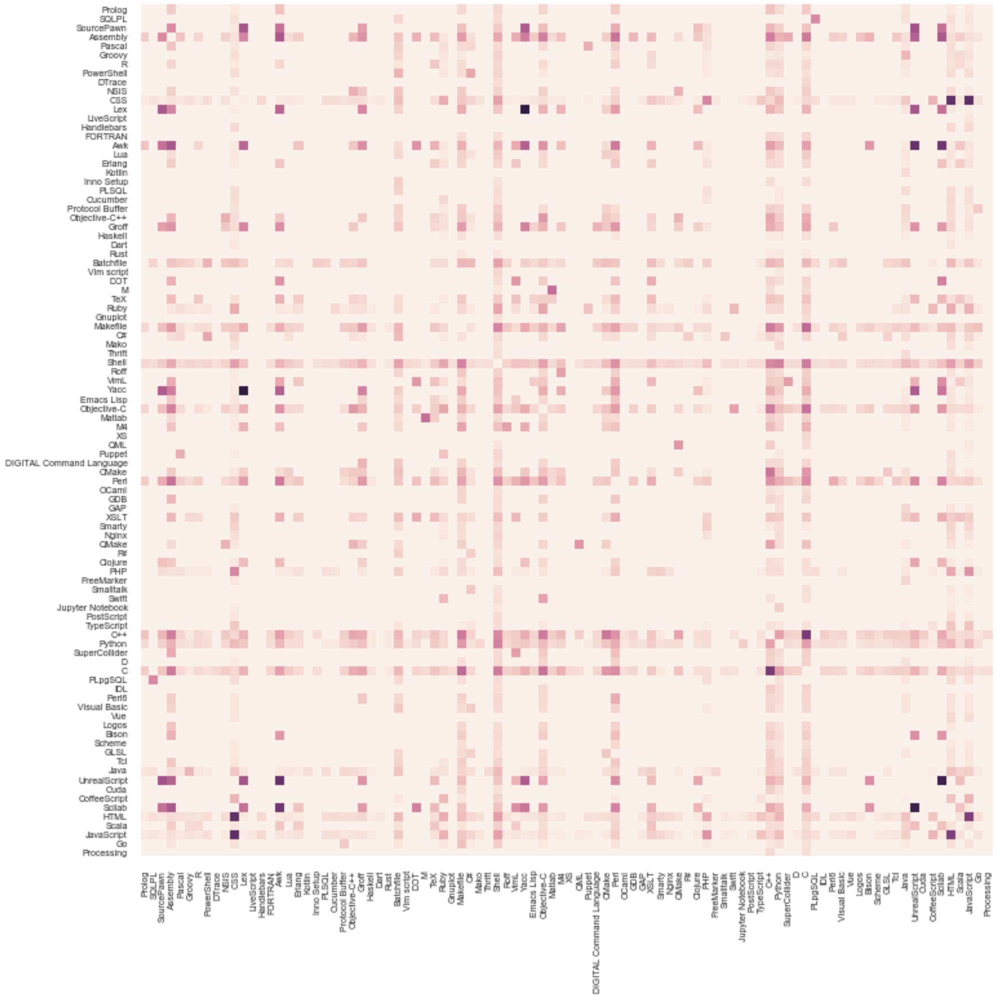


Use case: similarity matrices

- A similarity matrix quantifies how "similar" pairs of items are
- Similarity may be defined as cosine similarity for items associated with feature vectors: for two non-null feature vectors a and b in $\mathbf{R}^n$, the cosine similarity is

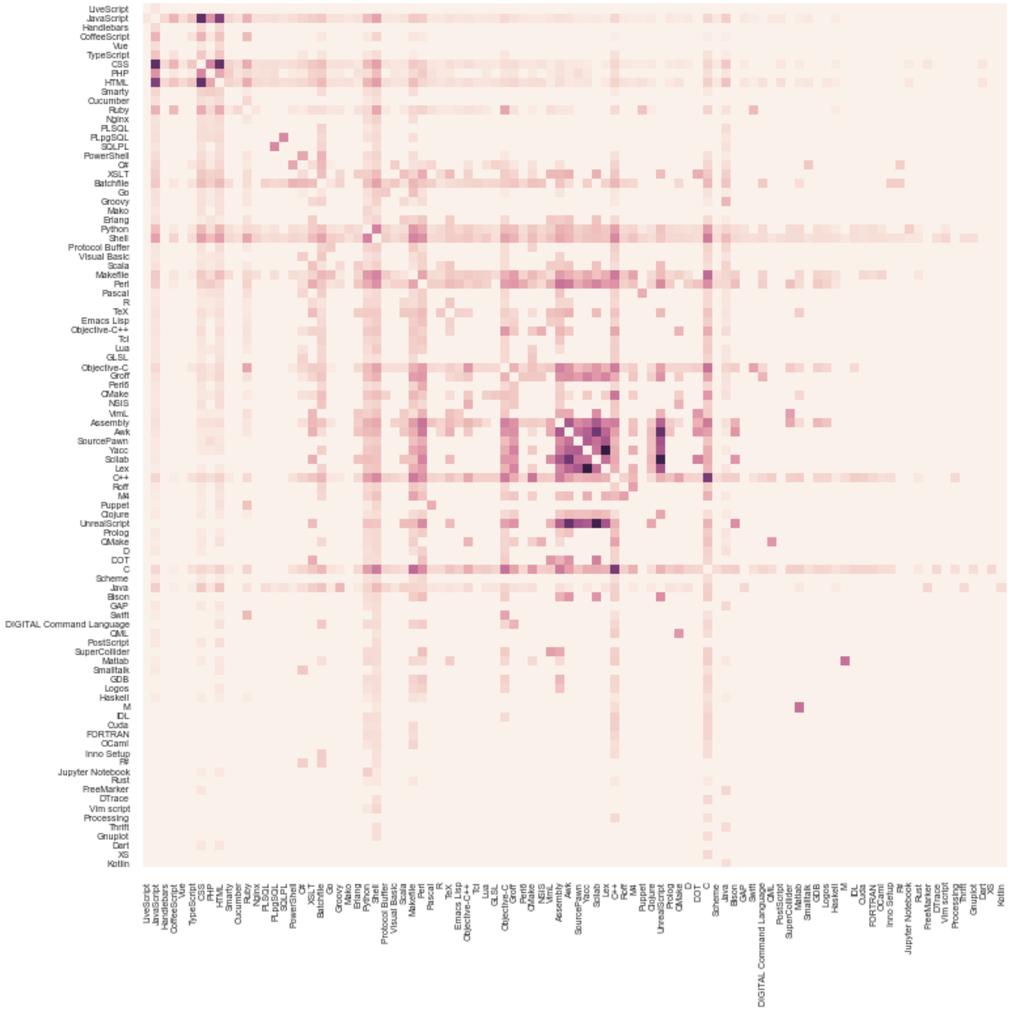
$$\mathrm{sim}(a,b) = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\ \sqrt{\sum_{i=1}^{n} b_i^2}}$$

- In our example, a feature vector associated with a programming language indicates its usage over different repositories



Use case: similarity matrices (cont'd)

- How can we order rows and columns of a matrix to visualize any possibly existing clusters in matrix data?

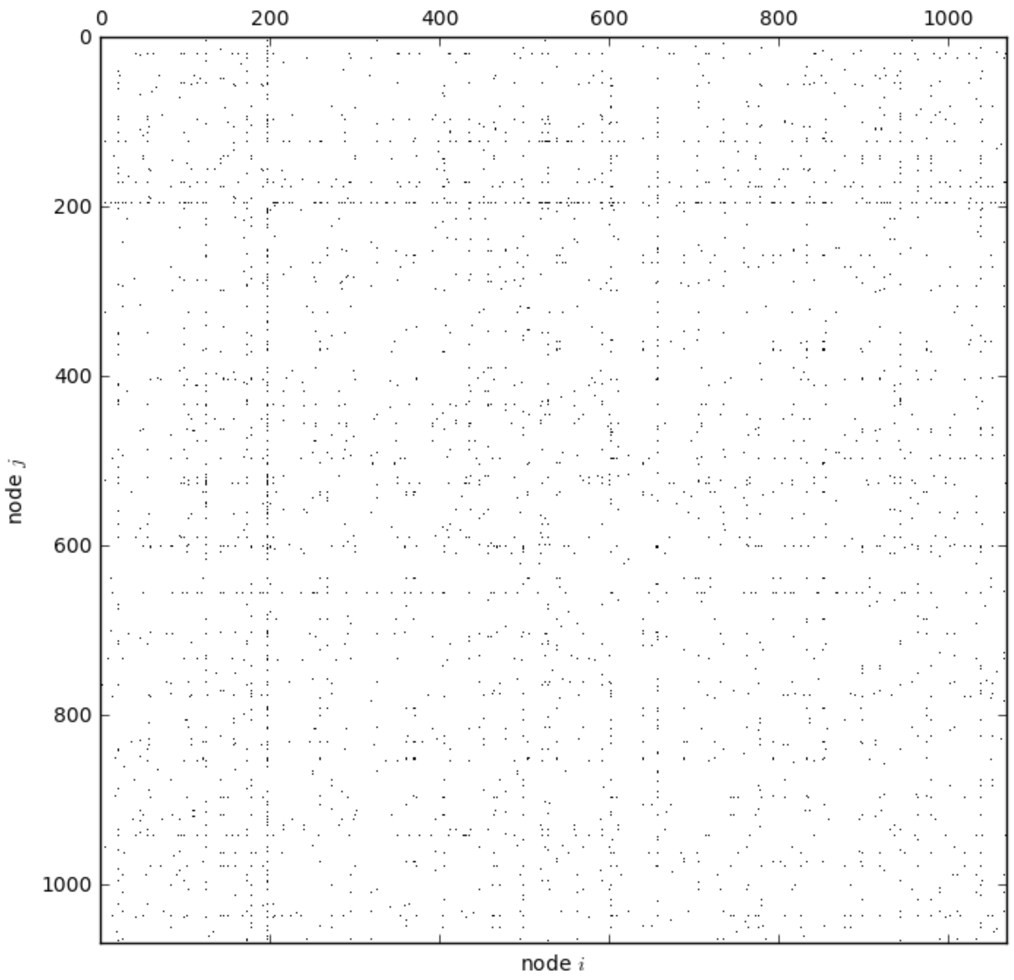


Use case: adjacency matrices

- Adjacency matrix of a graph indicates whether or not there is an edge between different pairs of nodes of the graph

- The plot in this slide shows the adjacency matrix of a graph that specifies existence of communications between Amazon Mechanical Turk workers

- Dataset source: Yin et al, The Communication Network within the Crowd, WWW 2016

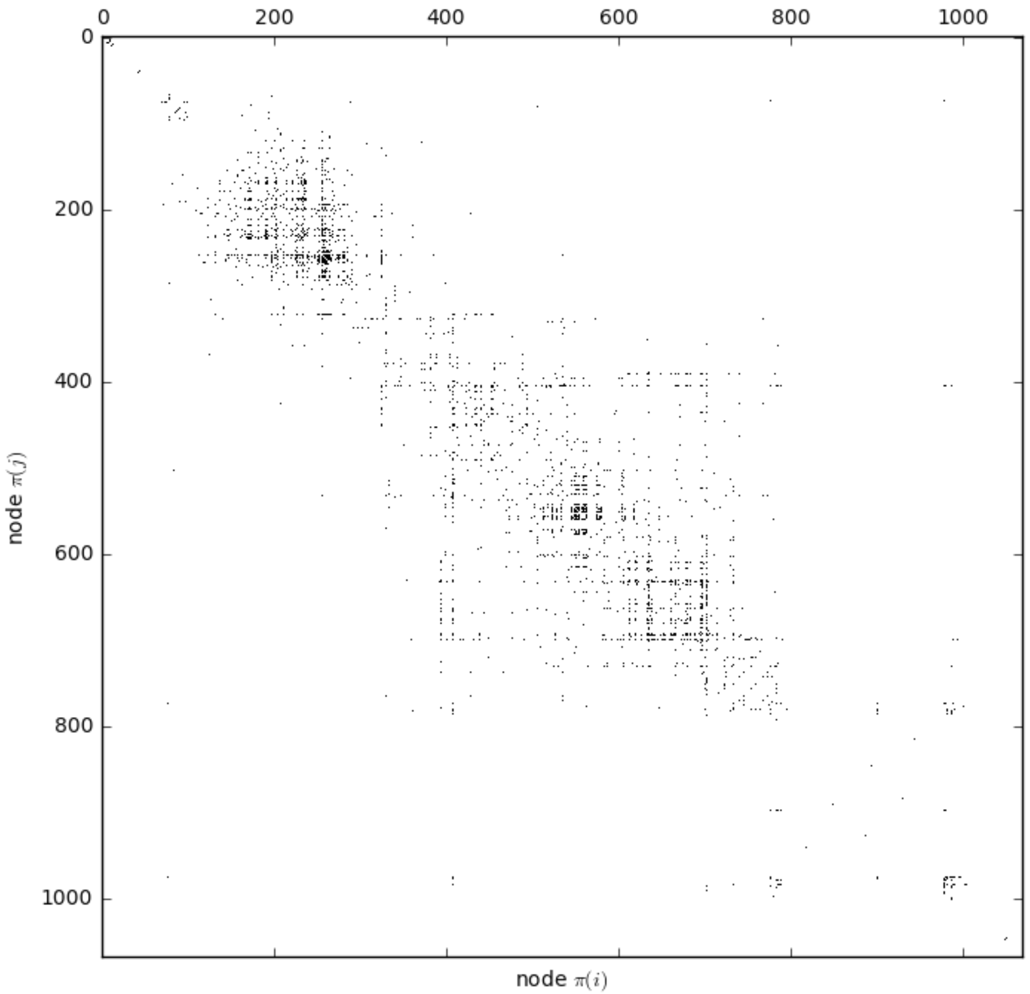


Use case: adjacency matrices

- Reordered rows and columns

  i.e. rearranged node identifiers

- The adjacency matrix with reordered rows and columns in this slide suggests existence of different communities of workers



Software modules

- Scikit-learn biclustering module
  - Scikit-learn Section 2.4 Biclustering

    http://scikit-learn.org/stable/modules/biclustering.html

  - from sklearn.cluster.bicluster import SpectralCoclustering
  - from sklearn.cluster.bicluster import SpectralBiclustering

- R seriation package
  - https://cran.r-project.org/web/packages/seriation/index.html

Some linear algebra concepts

Eigenvalues and eigenvectors

- λ is an eigenvalue of matrix A if for some vector $x \neq 0$

$$Ax = \lambda x$$

- A corresponding vector x is called an eigenvector

Laplacian matrix

- Let A be a real symmetric matrix

- The Laplacian matrix L_A is defined by

$$L_A = D_A - A$$

where D_A is a diagonal matrix with $d_{i,i} = \sum_{j=1}^{n} a_{i,j}$

- A is a real symmetric matrix $\Rightarrow L_A$ is a real symmetric matrix

Laplacian matrix (cont'd)

- For any real symmetrix matrix $A \in \mathbf{R}^{n \times n}$:

  - A has n eigenvectors
  - All eigenvectors of A are pairwise orthogonal
  - All eigenvalues of A are real

- Every Laplacian matrix has all its eigenvalues real and non-negative, which is equivalent to saying that it is positive semi-definite

- Every Laplacian matrix has the vector of all ones e as an eigenvector corresponding to the eigenvalue zero

Fiedler value and eigenvector

- For real symmetric matrix $A \in \mathbf{R}^{n \times n}$, Fiedler value is defined as the minimum eigenvalue of the Laplacian L_A that has an eigenvector orthogonal to e

- The corresponding eigenvector is called a Fiedler eigenvector

- The Fiedler value is the optimum value of the optimization problem:

$$
\begin{array}{rl}
\text{minimize} & x^T L_A x \\
\text{subject to} & x^T e = 0 \\
 & x^T x = 1
\end{array}
$$

Historical remarks

- Miroslav Fiedler
- Czech Republic’s mathematician
- 1926-2015
- Charles University, Prague

- M. Fiedler, Algebraic connectivity of graphs, Czehoslovak Math. J., 23(98), 1973

Laplacian matrix (cont'd)

- **Lemma**: For any real, symmetric matrix A:

$$x^T L_A x = \sum_{i<j} a_{i,j} \left(x_i - x_j\right)^2$$

- Laplacian matrices are closely related to graph cuts
  - Graph $G = (V, E)$ with edge weights: edge (i, j) has weight $a_{i,j}$
  - A is the adjacency matrix of G
  - Let x takes value in $\{-1,1\}^n$ defining a vertex cut: partitioning vertices into two sets (negative and positive labeled)
  - Graph cut is defined as the sum of weights of edges whose end vertices belong to different components of the vertex cut

Seriation

Seriation

- Input: a real, symmetric matrix $A \in \mathbf{R}^{n \times n}$
  - $a_{i,j}$ interpreted as the similarity between items i and j

- Goal: find a linear ordering (permutation) of items such that similar items are placed nearby, specifically, find a permutation π^* that minimizes

$$c(\pi) = \sum_{i<j} a_{i,j} \left(\pi_i - \pi_j\right)^2$$

over the set of all possible permutations of n elements Π_n

- This problem is NP hard

Fractional relaxation

- Find optimal solution to the following problem:

$$\begin{array}{rl} \text{minimize} & c(x) \\ \text{subject to} & x^T e = 0 \\ & x^T x = 1 \\ & x \in \mathbf{R}^n \end{array}$$

- The first constraint avoids multiplicity of solutions by adding a positive constant to each coordinate of x
- The second constraint avoids vector 0 to be a trivial solution
- The optimum solution is a Fiedler vector of A

Robinson matrices

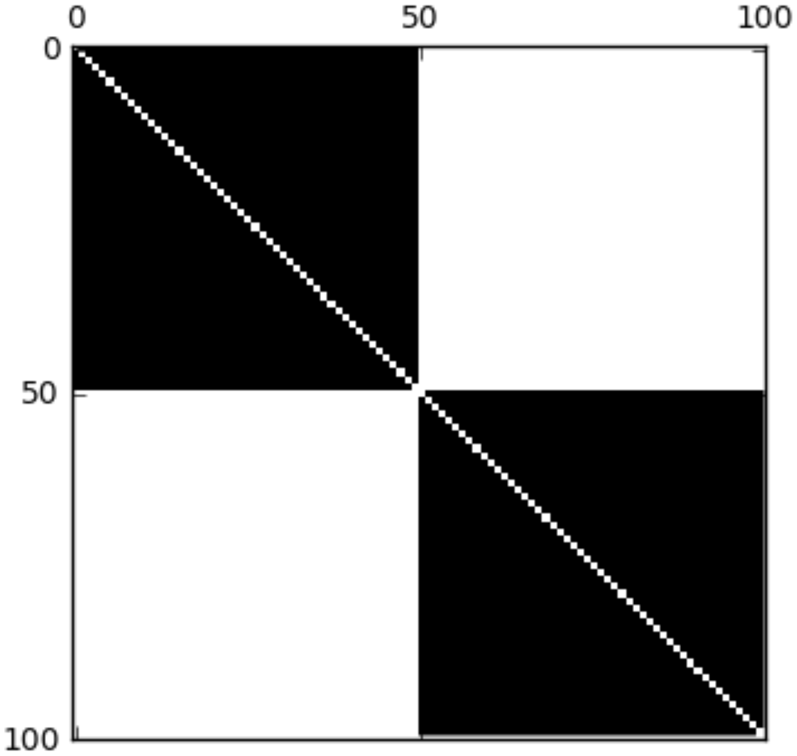
- A matrix A is said to be a Robinson matrix (R-matrix) if, and only if,
  - A is symmetric
  - $a_{i,j} \leq a_{i,k}$ for $j < k < i$ and $a_{i,j} \geq a_{i,k}$ for $i < j < k$

- In other words, a matrix is a R-matrix if it is symmetric and it has off-diagonal elements non-decreasing by moving away from the diagonal

- A matrix A is said to be a pre-R matrix if it can be symmetrically permuted (reordering rows and columns by the same permutation) to become an R-matrix

An example of a R-matrix

$$A = (a_{i,j})$$



- Clearly, a symmetric matrix
- The elements are decreasing as we move away from the diagonal along any row

black point if $a_{i,j} = 1$
white point if $a_{i,j} = 0$

An example R-matrix: stochastic block model

- Stochastic block model is a random graph model commonly studied in the community detection and graph clustering literature
- A stochastic block model can be defined by
  - Parameters $0 \leq q \leq p \leq 1$
  - Set of vertices $V = \{1,2, \ldots, n\}$
  - Hidden bipartition of vertices $(S, V \setminus S)$
  - Set of edges E: for every pair of vertices (i, j) such that $i < j$ we have

$$(i, j) \in E \text{ independently with probability} = \begin{cases} p & \text{if } i, j \in S \text{ or } i, j \in V \setminus S \\ q & \text{otherwise} \end{cases}$$

- A is defined as the adjacency matrix of G
- The case $q = 0$ and $p = 1$ is trivial when the vertex-set partition is not hidden

Fiedler vector of R-matrices

- **Theorem**: If A is a R-matrix then it has a monotone Fiedler vector.

- Proof: Atkins et al (1998)

Fiedler vector of R-matrices (cont'd)

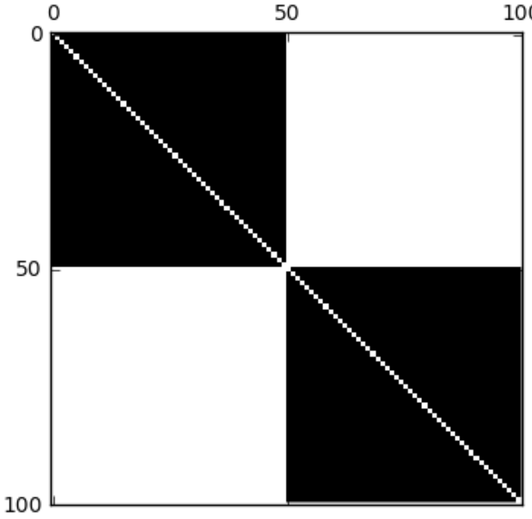
- **Theorem**: Let A be a pre-R matrix with a simple Fiedler value (i.e. unique value) and a Fiedler vector with no repeated elements. Let π_1 be the permutation sorting elements of the Fiedler vector in increasing order. Let π_2 be the permutation sorting elements of the Fiedler vector in decreasing order. Let Π_1 and Π_2 be the corresponding permutation matrices.

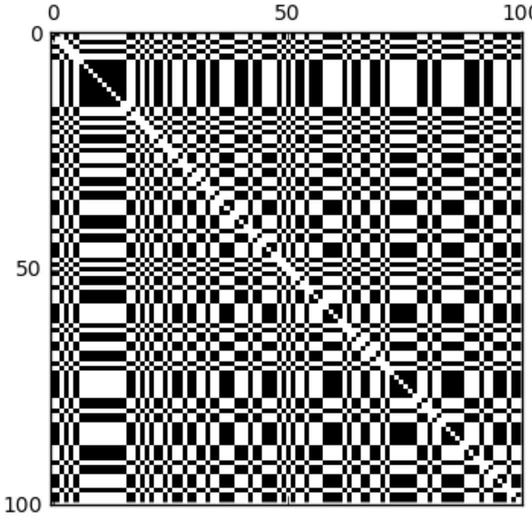
  Then, $\Pi_1 A \Pi_1$ and $\Pi_2 A \Pi_2$ are R-matrices and not other symmetric permutation of A produces an R matrix.

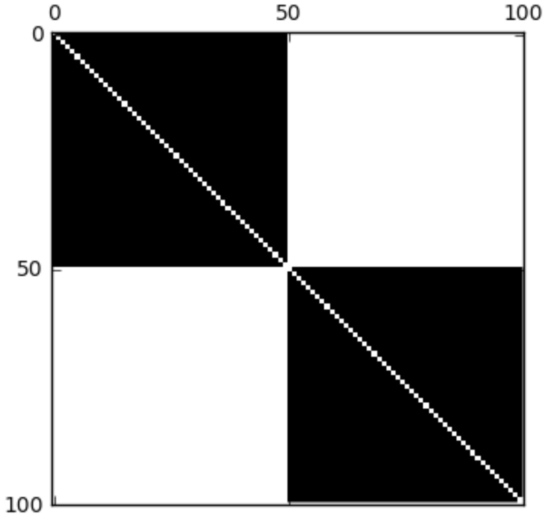
- Proof: Atkins et al (1998)

- See Thm 4.7 in Atkins et al for a statement under weaker assumptions

Example 1: p = 1, q = 0





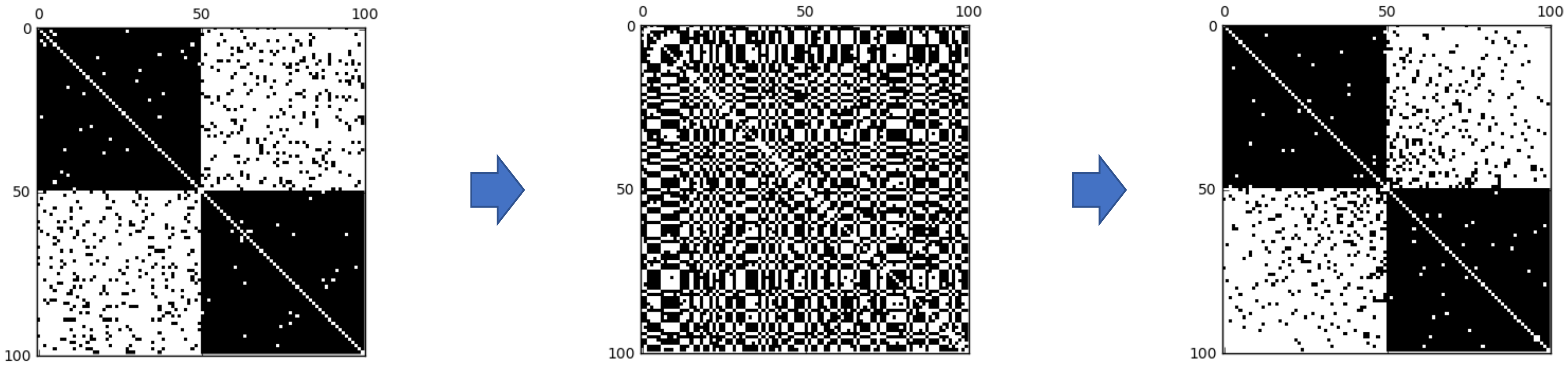


Rows and columns
permuted by the same
random permutation

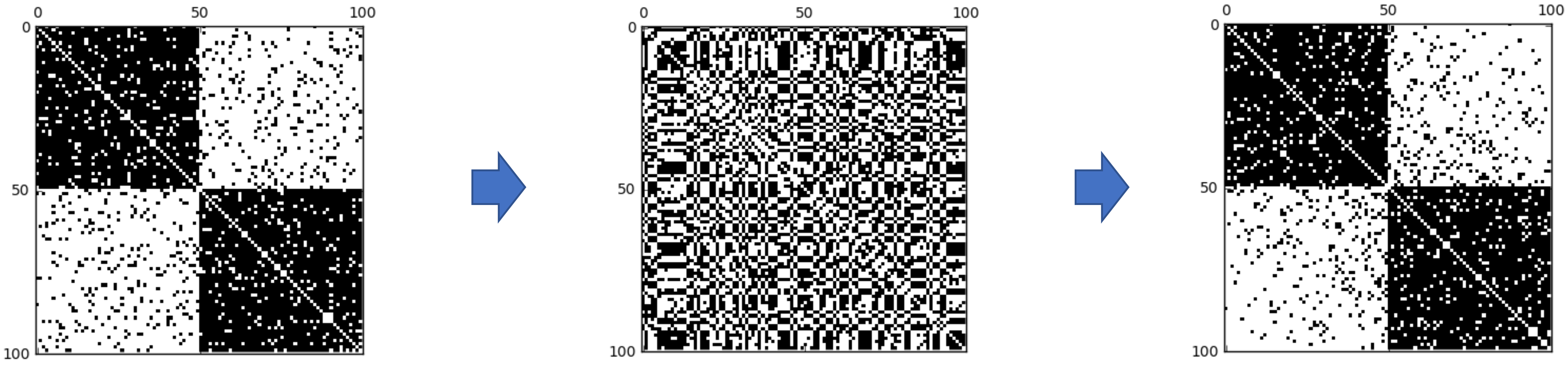
“symmetric permutation”

Rows and columns
sorted in a monotonic
order of the Fiedler
vector elements

Example 2: p = 0.99, q = 0.01



Example 3: p = 0.9, q = 0.1



Spectral co-clustering

Graph cuts

- Let V be a set of vertices with weights $w_{i,j}$ for $i, j \in V$

- k-way partition: let $P_k(S)$ be the set of all possible partitions of S in k components
  - In particular, $(S_1, S_2) \in P_2(S)$ is referred as a bipartition

- The cut function is defined as the sum of weights of cut edges:

$$\text{cut}(V_1, V_2) = \sum_{i \in V_1, j \in V_2} w_{i,j}$$

- More generally, we define

$$\text{cut}(V_1, V_2, \dots, V_k) = \sum_{1 \le i < j \le k} \text{cut}(V_i, V_j)$$

Graph cuts and the Laplacian matrix

- A bipartition (V_1, V_2) can be represented by a partition vector x defined as

$$x_i = \begin{cases} 1 & \text{if } i \in V_1 \\ -1 & \text{if } i \in V_2 \end{cases}$$

- **Lemma**: Any bipartition (V_1, V_2) and the corresponding partition vector x satisfy

$$\frac{x^T L_A x}{x^T x} = \frac{1}{n} 4 \operatorname{cut}(V_1, V_2)$$

Normalized cut function

- Define the weight of a set of vertices S: $w(S) = \sum_{i \in S} w_i$

- The normalized cut function is defined as follows:

$$Q(V_1, V_2) = \frac{\mathrm{cut}(V_1, V_2)}{w(V_1)} + \frac{\mathrm{cut}(V_1, V_2)}{w(V_2)}$$

- This cut function captures *both* sparsity of edge cuts and balancing of the component sizes

- Optimization problem formulation: minimize $Q(V_1, V_2)$ over $P_2(V)$

Special normalized cut functions

- **Ratio-cut**
  - Each vertex has a unit weight
  - Amounts to looking at the eigenvalue problem: $L_A y = \lambda y$

- **Normalized-cut**
  - Each vertex weight is equal to the sum of weights of incident edges
  - In this case $w(V_i) = \text{cut}(V_1, V_2) + w^{int}(V_i)$
    where $w^{int}(V_i) = \sum_{(u,v)\in E: u,v \in V_i} w_{u,v}$
  - Amounts to the generalized eigenvalue problem : $L_A y = \lambda D_A y$

Normalized-cut function

- Show that

$$Q(V_1, V_2) = 2 - \left(\frac{w^{int}(V_1)}{w(V_1)} + \frac{w^{int}(V_2)}{w(V_2)} \right)$$

- Minimizing the normalized-cut is equivalent to maximizing the proportion of edge weights that lie within each component of a vertex set partition

Generalized partition-vector representation

- Let y be a generalized partition vector defined as

$$y_i = \begin{cases} \sqrt{\dfrac{w(V_2)}{w(V_1)}} & if\ i \in V_1 \\ -\sqrt{\dfrac{w(V_1)}{w(V_2)}} & if\ i \in V_2 \end{cases}$$

- **Properties**: (a) $y^T D_w y = w(V)$ and (b) $y^T D_w e = 0$

- **Lemma**:

$$\frac{y^T L_A y}{y^T D_w y} = \frac{\mathrm{cut}(V_1, V_2)}{w(V_1)} + \frac{\mathrm{cut}(V_1, V_2)}{w(V_2)}$$

Fractional relaxation

$$\text{minimize} \quad \frac{y^T L_A y}{y^T D_W y}$$

$$\begin{aligned} \text{subject to} \quad & y^T D_W e = 0 \\ & y \neq 0 \\ & y \in \mathbf{R}^n \end{aligned}$$

- **Theorem**: The solution is the eigenvector corresponding to the second smallest eigenvalue λ_2 of the generalized eigenvalue problem:

$$L_A y = \lambda D_W y$$

- **Corollary**: The optimum value of the min normalized cut is $\geq \lambda_2$

Bipartite graph clustering

- Bipartite graph
  - The vertex set consists of disjoint sets of vertices L and R
  - Each edge has its vertices in L and R

- Let D_L and D_R be diagonal matrices with diagonal elements

$$(D_L)_{i,i} = \sum_{j \in R} w_{i,j} \text{ and } (D_R)_{i,i} = \sum_{j \in L} w_{j,i}$$

- For graph G:

$$A = \begin{pmatrix} 0 & W \\ W^T & 0 \end{pmatrix}, D_A = \begin{pmatrix} D_L & 0 \\ 0 & D_R \end{pmatrix} \text{ and } L_A = \begin{pmatrix} D_L & -W \\ -W^T & D_R \end{pmatrix}$$

The generalized eigenvalue problem

- The generalized eigenvalue problem can be written as

$$D_L x - Wy = \lambda D_L x$$
$$-A^T x + D_R y = \lambda D_R y$$

- **Assumption**: W has a strictly positive element in each row and column

- Note that we can write:

$$D_L^{1/2} x - D_L^{-1/2} Wy = \lambda D_L^{1/2} x$$
$$-D_R^{-1/2} A^T x + D_R^{1/2} y = \lambda D_R^{1/2} y$$

Change of variables

- Using the change of variables

$$\widetilde{W} = D_L^{-1/2} W D_R^{-1/2}$$

$$u = D_L^{1/2} x$$

$$v = D_R^{1/2} y \text{ and}$$

$$\sigma = 1 - \lambda$$

we can write

$$\widetilde{W} v = \sigma u \quad \text{and} \quad \widetilde{W}^T u = \sigma v$$

Eigenvectors

- The eigenvector x_2 corresponding to the second smallest eigenvalue λ_2 of the generalized eigenvalue problem can be written as:

$$x_2 = \begin{pmatrix} D_L^{-1/2} u_2 \\ D_R^{-1/2} v_2 \end{pmatrix}$$

where u_2 and v_2 are the left and right singular vectors of $\widetilde{W}$ corresponding to the singular value $\sigma_2 = 1 - \lambda_2$

- We can think of u_2 to give a partition of the set of left vertices and v_2 to give a partition of the set of right vertices

Bi-clustering algorithm

- Input: W
- Compute $\widetilde{W} = D_L^{-1/2} W D_R^{-1/2}$
- Compute the left and right singular vectors u_2 and v_2 of $\widetilde{W}$ corresponding to the singular value $\sigma_2 = 1 - \lambda_2$
- Partition the set of vertices in two components using k-means algorithm for input data points x_2

k-way clustering algorithm

- Given a positive integer $k \geq 2$ the goal is to partition the set of left vertices and the set of right vertices in k components
- Let $U = (u_2, u_3, \ldots, u_{\ell+1})$ and $V = (v_2, v_3, \ldots, v_{\ell+1})$ be ℓ the left and right singular vectors
- Let $(x_2, x_3, \ldots, x_{\ell+1}) = \begin{pmatrix} D_L^{-1/2} U \\ D_R^{-1/2} V \end{pmatrix}$
- Apply the k-means algorithm to the input ℓ-dimensional points

Evaluating a bi-clustering: consensus score

- The definition of the consensus score as implemented in sklearn.metrics.consensus_score

- Quantifies similarity between two input *sets* of biclusters

- Similarity between individual biclusters is computed
  - Default is Jaccard similarity for two sets A and B defined as $|A \cap B|/|A \cup B|$

- The best matching between sets is found using the Hungarian algorithm

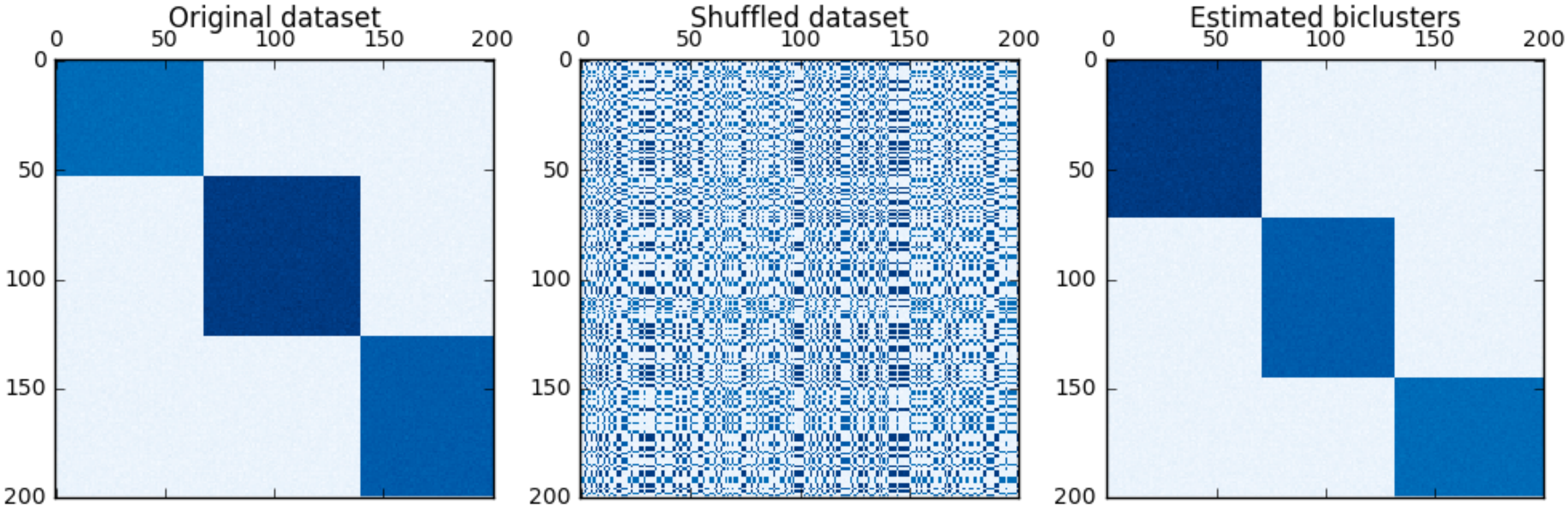
- The final score is the sum of the similarities divided by the size of the larger set

```
def consensus_score(a, b, similarity="jaccard"):

    Parameters
    ----------
    a : (rows, columns)
        Tuple of row and column indicators for a set of biclusters.

    b : (rows, columns)
        Another set of biclusters like ``a``.

    similarity : string or function, optional, default: "jaccard"
        May be the string "jaccard" to use the Jaccard coefficient, or
        any function that takes four arguments, each of which is a 1d
        indicator vector: (a_rows, a_columns, b_rows, b_columns).


    if similarity == "jaccard":
        similarity = _jaccard
    matrix = _pairwise_similarity(a, b, similarity)
    indices = linear_assignment(1. - matrix)
    n_a = len(a[0])
    n_b = len(b[0])
    return matrix[indices[:, 0], indices[:, 1]].sum() / max(n_a, n_b)
```

maximum weight matching (input is a cost matrix) from sklearn.utils.linear_assignment_

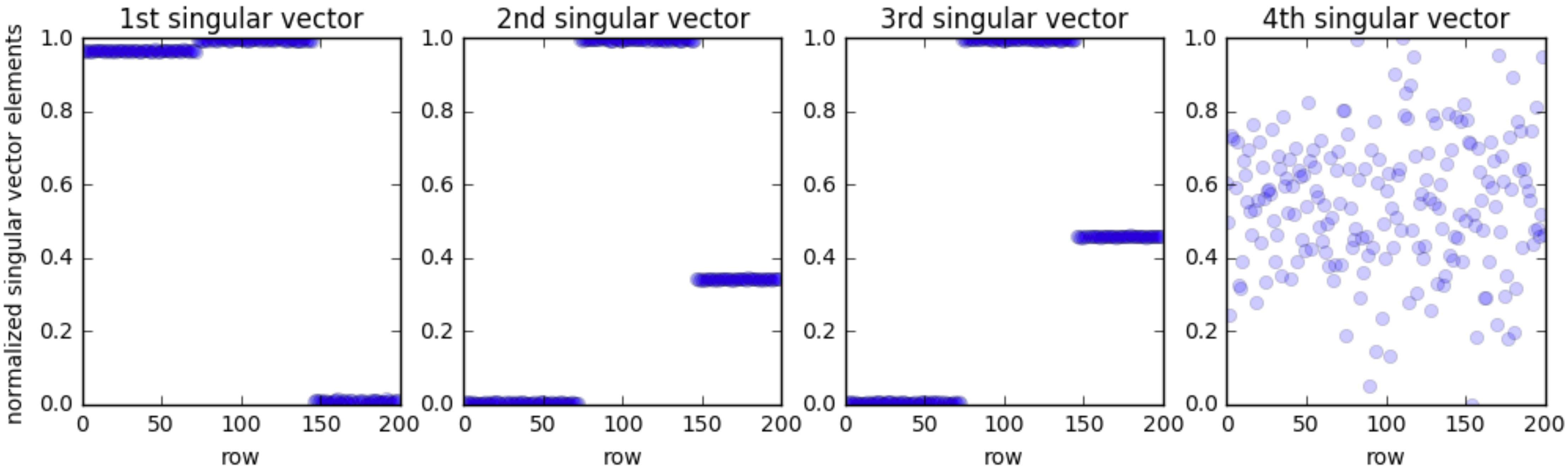
```python
def _jaccard(a_rows, a_cols, b_rows, b_cols):
    """Jaccard coefficient on the elements of the two biclusters."""
    intersection = ((a_rows * b_rows).sum() *
                    (a_cols * b_cols).sum())

    a_size = a_rows.sum() * a_cols.sum()
    b_size = b_rows.sum() * b_cols.sum()

    return intersection / (a_size + b_size - intersection)

def _pairwise_similarity(a, b, similarity):
    """Computes pairwise similarity matrix.

    result[i, j] is the Jaccard coefficient of a's bicluster i and b's
    bicluster j.

    """
    a_rows, a_cols, b_rows, b_cols = _check_rows_and_columns(a, b)        # unpacks rows and columns
    n_a = a_rows.shape[0]
    n_b = b_rows.shape[0]
    result = np.array(list(list(similarity(a_rows[i], a_cols[i],
                                           b_rows[j], b_cols[j])
                                for j in range(n_b))
                           for i in range(n_a)))
    return result
```

Example



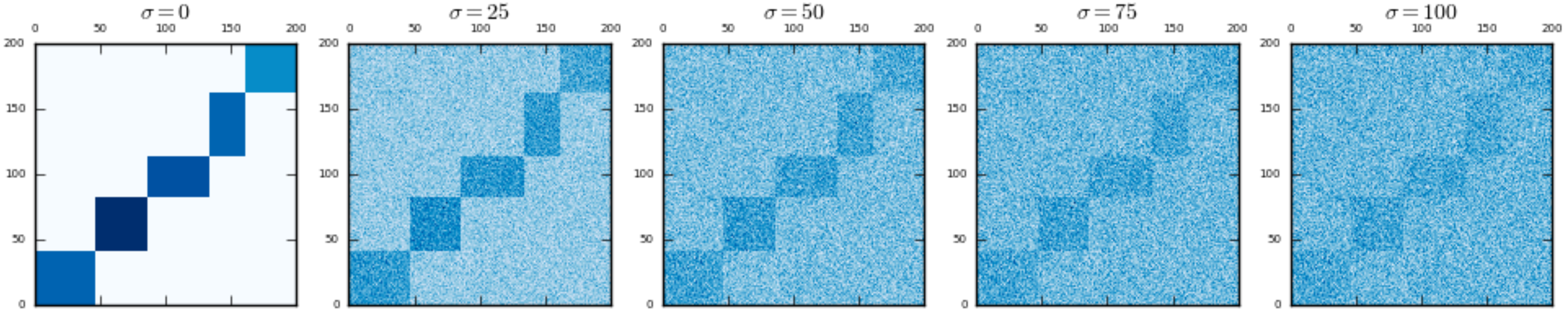
- Recovery of hidden co-clusters by spectral co-clustering

Example: singular vectors



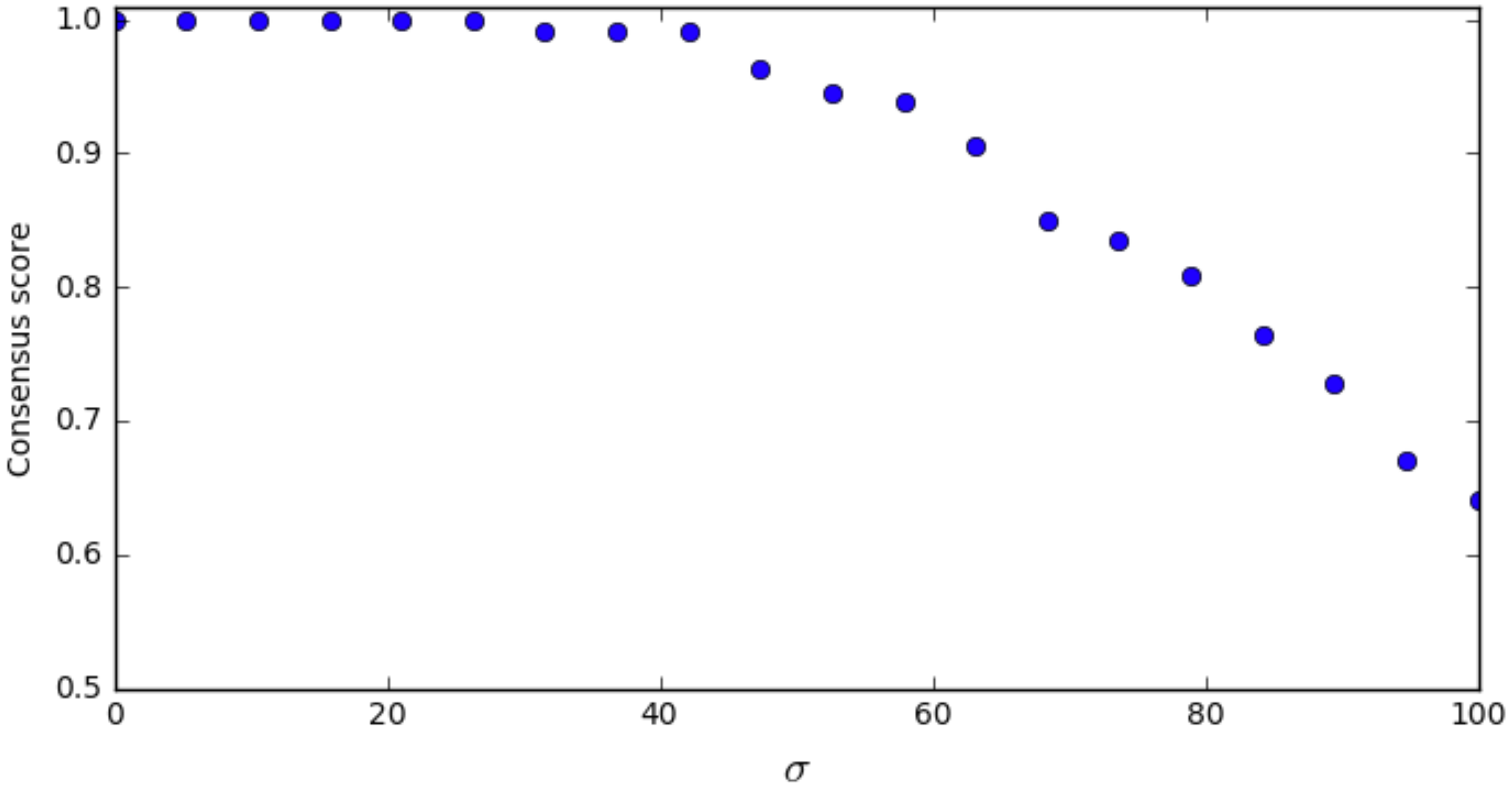
- Clusters are clearly indicated by the 2nd and 3rd singular vectors

Robustness to noise



- Input matrix corrupted by noise with varying variance σ

Robustness to noise (cont'd)



- Consensus score vs standard deviation of noise

References

- J. E. Atkins, E. G. Boman and B. Hendrickson, A Spectral Algorithm for Seriation and the Consecutive Ones Problem, SIAM J. Computing, Vol 28, No 1, pp. 297-310, 1998
- Y. Benjamini, Opening the Box of a Boxplot, The American Statistician, Vol 42, No 4, pp 257-262, 1988
- I. S. Dhilon, Co-clustering documents and words using bipartite spectral graph partitioning, Proc. of ACM KDD, pp. 269-274, 2001
- M. Eisen, P. Spellman, P. Brown and D. Botstein, Cluster analysis and display and display of genome-wide expression patterns, PNAS, Vol 95, pp 14863-68, 1998
- M. Friendly, Corrgrams: exploratory displays for correlation matrices, The American Statistician, Vol 56, pp. 316-324, 2002
- Y. Kluger, R. Basri, J. T. Chang, M. Gerstein, Spectral Biclustering of Microarray Data: Coclustering Genes and Conditions. Genome Research, 13(4), pp. 703-716 , 2003
- L. Wilkinson and M. Friendly, The history of the cluster heat map, The American Statistician, Vol 63, No 2, pp 179-184, 2009