Statistical analysis of network data lecture 9

Sofia Olhede

November 13, 2025

Introduction to Biclustering

- Many datasets have a two-way structure where we observe responses across two dimensions (e.g., users $\times$ products, patients $\times$ genes).
- **Standard clustering** operates on one dimension at a time, either cluster rows *or* columns, but not both.
- **Biclustering** simultaneously clusters *both* dimensions, finding subgroups of rows and columns that exhibit similar patterns.

Introduction to Biclustering

- Many datasets have a two-way structure where we observe responses across two dimensions (e.g., users $\times$ products, patients $\times$ genes).
- **Standard clustering** operates on one dimension at a time, either cluster rows *or* columns, but not both.
- **Biclustering** simultaneously clusters *both* dimensions, finding subgroups of rows and columns that exhibit similar patterns.
- Say we have an $n \times m$ matrix X where rows and columns represent different entity types.
- We want to identify rectangular blocks within the matrix where entries display similar behavior.

Applications of Biclustering

Examples of biclustering applications include:

- For online retail, X_{ij} can then show if user i wants product j, and our task is to segment users and products into relevant subgroups. Who might want product j ?
- In bioinformatics, X_{ij} could correspond to the log activation level of gene j in patient i. Our task is then to determine groups of patients with similar genetic profiles, while at the same time finding groups of genes with similar activation levels.
- In medicine, determining groups in such data bases has helped to identify associations between active ingredients and adverse medical reactions.

Some notation

Recall we have some data observed in a matrix X of size n by m.

We will assign each row and column a name $R = \{r_1, \ldots, r_n\}$ and $C = \{c_1, \ldots, c_m\}$ respectively.

- We do this because we often do not know the ordering of the rows and columns, and it is often useful to permute them.
- In practice, these might correspond to users and products, patients and genes, etc.

We also have some indexing notation:

- X_{ij} denotes the ijth element of X, where $i \in R$ and $j \in C$.
- X_{IJ} denotes the submatrix that has rows $I \subseteq R$ and columns $J \subseteq C$.

Row and Column Clusters

We define a **row cluster** as a subset $I \subseteq R$ of rows that exhibit similar behaviour across all columns.

- A row cluster is denoted (I, C), where $I \subseteq R$.
- The corresponding data submatrix X_{IC} has $|I|$ rows and m columns.

Row and Column Clusters

We define a **row cluster** as a subset $I \subseteq R$ of rows that exhibit similar behaviour across all columns.

- A row cluster is denoted (I, C), where $I \subseteq R$.
- The corresponding data submatrix X_{IC} has $|I|$ rows and m columns.

A **column cluster**, in contrast, is a subset $J \subseteq C$ of columns that exhibit similar behaviour across all rows.

- A column cluster is denoted (R, J), where $J \subseteq C$.
- The corresponding data submatrix X_{RJ} has n rows and $|J|$ columns.

Definition of Biclusters

In contrast, a **bicluster** is a subset of rows that exhibit similar behaviour across a subset of columns, and vice versa.

- A bicluster is denoted (I, J) where $I \subseteq R$ and $J \subseteq C$.
- The corresponding data submatrix X_{IJ} has $|I|$ rows and $|J|$ columns.

Given a data matrix X we want to identify a set of biclusters $B_k = (I_k, J_k)$ so that B_k is in some sense homogeneous.

Data Matrices as Bipartite Graphs

- A data matrix can be viewed as a bipartite graph.
- Recall, a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, is said to be **bipartite** if its vertices can be partitioned into two sets L and U such that every edge in E has exactly one end in L and the other element in U. Furthermore $V = L \cup U$.
- For a data matrix X, we can construct a bipartite graph where $L = R$ (rows), $U = C$ (columns), and edges represent non-zero entries or relationships in X.
- Biclustering then corresponds to finding dense subgraphs in this bipartite representation.

Types of Biclusters

We now distinguish different types of biclusters based on the patterns in their corresponding data submatrices:

1. **Constant values:** for all $i \in I$ and $j \in J$

$$X_{ij} \approx \mu$$

Types of Biclusters

We now distinguish different types of biclusters based on the patterns in their corresponding data submatrices:

1. **Constant values:** for all $i \in I$ and $j \in J$

$$X_{ij} \approx \mu$$

2. **Constant values on rows or columns:** for all $i \in I$ and $j \in J$

$$X_{ij} \approx \mu + \alpha_i \text{ or } X_{ij} \approx \mu + \beta_j$$

$$X_{ij} \approx \mu \times \alpha_i \text{ or } X_{ij} \approx \mu \times \beta_j$$

Types of Biclusters

We now distinguish different types of biclusters based on the patterns in their corresponding data submatrices:

1. **Constant values:** for all $i \in I$ and $j \in J$

$$X_{ij} \approx \mu$$

2. **Constant values on rows or columns:** for all $i \in I$ and $j \in J$

$$X_{ij} \approx \mu + \alpha_i \text{ or } X_{ij} \approx \mu + \beta_j$$

$$X_{ij} \approx \mu \times \alpha_i \text{ or } X_{ij} \approx \mu \times \beta_j$$

3. **Coherent values:** for all $i \in I$ and $j \in J$

$$X_{ij} \approx \mu + \alpha_i + \beta_j \text{ or } X_{ij} \approx \mu \times \alpha_i \times \beta_j$$

Types of Biclusters

We now distinguish different types of biclusters based on the patterns in their corresponding data submatrices:

1. **Constant values:** for all $i \in I$ and $j \in J$

$$X_{ij} \approx \mu$$

2. **Constant values on rows or columns:** for all $i \in I$ and $j \in J$

$$X_{ij} \approx \mu + \alpha_i \text{ or } X_{ij} \approx \mu + \beta_j$$

$$X_{ij} \approx \mu \times \alpha_i \text{ or } X_{ij} \approx \mu \times \beta_j$$

3. **Coherent values:** for all $i \in I$ and $j \in J$

$$X_{ij} \approx \mu + \alpha_i + \beta_j \text{ or } X_{ij} \approx \mu \times \alpha_i \times \beta_j$$

4. **Coherent evolutions:** Entries preserve ordering/trends across rows or columns

Types of Biclusters

We now distinguish different types of biclusters based on the patterns in their corresponding data submatrices:

1. **Constant values:** for all $i \in I$ and $j \in J$

$$X_{ij} \approx \mu$$

2. **Constant values on rows or columns:** for all $i \in I$ and $j \in J$

$$X_{ij} \approx \mu + \alpha_i \text{ or } X_{ij} \approx \mu + \beta_j$$

$$X_{ij} \approx \mu \times \alpha_i \text{ or } X_{ij} \approx \mu \times \beta_j$$

3. **Coherent values:** for all $i \in I$ and $j \in J$

$$X_{ij} \approx \mu + \alpha_i + \beta_j \text{ or } X_{ij} \approx \mu \times \alpha_i \times \beta_j$$

4. **Coherent evolutions:** Entries preserve ordering/trends across rows or columns

The first three types analyse the observed values in X; the fourth tries to identify coherent behaviour.

Summary Statistics for Biclusters

To numerically summarize the data matrix we define:

- The mean of row i over columns J:

$$\bar{x}_{i,J} = \frac{1}{|J|} \sum_{j \in J} x_{ij}$$

Summary Statistics for Biclusters

To numerically summarize the data matrix we define:

- The mean of row i over columns J:

$$\bar{x}_{i,J} = \frac{1}{|J|} \sum_{j \in J} x_{ij}$$

- The mean of column j over rows I:

$$\bar{x}_{I,j} = \frac{1}{|I|} \sum_{i \in I} x_{ij}$$

- The overall mean of the bicluster (I, J):

$$\bar{x}_{I,J} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} x_{ij}$$

Constant Biclusters and Hartigan's Method

A **perfect constant bicluster** is a submatrix (I, J) where all values are equal:

$$X_{ij} = \mu \quad \text{for all } i \in I, j \in J$$

In practice, with noise, we observe:

$$X_{ij} = \mu + \eta_{ij}$$

where η_{ij} represents measurement noise or random variation.

Constant Biclusters and Hartigan's Method

A **perfect constant bicluster** is a submatrix (I, J) where all values are equal:

$$X_{ij} = \mu \quad \text{for all } i \in I, j \in J$$

In practice, with noise, we observe:

$$X_{ij} = \mu + \eta_{ij}$$

where η_{ij} represents measurement noise or random variation.

Hartigan's algorithm finds constant biclusters by:

- Partitioning the data matrix into submatrices (biclusters)
- Evaluating bicluster quality using within-bicluster variance:

Constant Biclusters and Hartigan's Method

A **perfect constant bicluster** is a submatrix (I, J) where all values are equal:

$$X_{ij} = \mu \quad \text{for all } i \in I, j \in J$$

In practice, with noise, we observe:

$$X_{ij} = \mu + \eta_{ij}$$

where η_{ij} represents measurement noise or random variation.

Hartigan's algorithm finds constant biclusters by:

- Partitioning the data matrix into submatrices (biclusters)
- Evaluating bicluster quality using within-bicluster variance:

$$\widehat{\text{Var}}(X_{IJ}) = \sum_{i \in I, j \in J} \left(X_{ij} - \bar{X}_{I,J}\right)^2$$

Lower variance indicates a more homogeneous (better) bicluster.

Hartigan's Partitioning Algorithm

Hartigan's method uses a divisive (top-down) approach to partition the matrix.

Starting point: The entire matrix as one bicluster.

Hartigan's Partitioning Algorithm

Hartigan's method uses a divisive (top-down) approach to partition the matrix.

Starting point: The entire matrix as one bicluster.

Iterative splitting: At each step, greedily choose either:

- A row split: partition some row cluster I into I_1 and I_2, OR
- A column split: partition some column cluster J into J_1 and J_2

Choose the split that gives the largest reduction in total within-bicluster variance.

Hartigan's Partitioning Algorithm

Hartigan's method uses a divisive (top-down) approach to partition the matrix.

Starting point: The entire matrix as one bicluster.

Iterative splitting: At each step, greedily choose either:

- A row split: partition some row cluster I into I_1 and I_2, OR
- A column split: partition some column cluster J into J_1 and J_2

Choose the split that gives the largest reduction in total within-bicluster variance.

Objective: Minimize total variance across all biclusters:

$$\sum_{\text{all biclusters } (I,J)} \sum_{r \in I, c \in J} \left(X_{rc} - \bar{X}_{I,J}\right)^2$$

Stop when adding more splits doesn't sufficiently reduce variance (using permutation tests).

Biclusters with Constant Rows or Columns

A perfect bicluster with constant columns is a submatrix (I, J), where all the values within the bicluster can be obtained using one of the following expressions:

$$X_{ij} = \mu + \alpha_i$$
$$X_{ij} = \mu \times \alpha_i.$$

Here μ is the typical value within the bicluster and α_i is the adjustment for row $i \in I$. A perfect bicluster with constant rows is analagous.

- If searching for multiple biclusters, the row/column effects α_i and β_j must be the same across all biclusters.
- This allows preprocessing: rescale rows (or columns) to remove effects, then apply constant bicluster methods to the rescaled matrix.

Coherent Value Biclusters: Additive Model

A **perfect coherent bicluster** (I, J) has an additive structure with both row and column effects:

$$X_{ij} = \mu + \alpha_i + \beta_j \quad \text{for all } i \in I, j \in J$$

Coherent Value Biclusters: Additive Model

A **perfect coherent bicluster** (I, J) has an additive structure with both row and column effects:

$$X_{ij} = \mu + \alpha_i + \beta_j \quad \text{for all } i \in I, j \in J$$

where:

- μ is the baseline value for the bicluster
- α_i is the row effect for row $i \in I$
- β_j is the column effect for column $j \in J$

Coherent Value Biclusters: Additive Model

A **perfect coherent bicluster** (I, J) has an additive structure with both row and column effects:

$$X_{ij} = \mu + \alpha_i + \beta_j \quad \text{for all } i \in I, j \in J$$

where:

- μ is the baseline value for the bicluster
- α_i is the row effect for row $i \in I$
- β_j is the column effect for column $j \in J$

This extends Type 2 biclusters (constant rows/columns) by allowing *both* row and column effects simultaneously.

Perfect Coherent Biclusters

In a perfect coherent bicluster, each entry X_{ij} can be exactly reconstructed from summary statistics:

$$X_{ij} = \bar{X}_{i,J} + \bar{X}_{I,j} - \bar{X}_{I,J}$$

Perfect Coherent Biclusters

In a perfect coherent bicluster, each entry X_{ij} can be exactly reconstructed from summary statistics:

$$X_{ij} = \bar{X}_{i,J} + \bar{X}_{I,j} - \bar{X}_{I,J}$$

To verify this, note:

$$\begin{aligned}
\bar{X}_{i,J} &= \mu + \alpha_i + \bar{\beta}_J \\
\bar{X}_{I,j} &= \mu + \bar{\alpha}_I + \beta_j \\
\bar{X}_{I,J} &= \mu + \bar{\alpha}_I + \bar{\beta}_J
\end{aligned}$$

Perfect Coherent Biclusters

In a perfect coherent bicluster, each entry X_{ij} can be exactly reconstructed from summary statistics:

$$X_{ij} = \bar{X}_{i,J} + \bar{X}_{I,j} - \bar{X}_{I,J}$$

To verify this, note:

$$\begin{aligned}\bar{X}_{i,J} &= \mu + \alpha_i + \bar{\beta}_J \\ \bar{X}_{I,j} &= \mu + \bar{\alpha}_I + \beta_j \\ \bar{X}_{I,J} &= \mu + \bar{\alpha}_I + \bar{\beta}_J\end{aligned}$$

Therefore:

$$\begin{aligned}\bar{X}_{i,J} + \bar{X}_{I,j} - \bar{X}_{I,J} &= (\mu + \alpha_i + \bar{\beta}_J) + (\mu + \bar{\alpha}_I + \beta_j) - (\mu + \bar{\alpha}_I + \bar{\beta}_J) \\ &= \mu + \alpha_i + \beta_j\end{aligned}$$

Mean Squared Residue (Cheng & Church, 2000)

In practice, biclusters are not perfect. Define the **residue** for entry (i, j):

$$r(X_{ij}) = X_{ij} - \bar{X}_{i,J} - \bar{X}_{I,j} + \bar{X}_{I,J}$$

This measures deviation from the perfect additive structure.

Mean Squared Residue (Cheng & Church, 2000)

In practice, biclusters are not perfect. Define the **residue** for entry (i, j):

$$r(X_{ij}) = X_{ij} - \bar{X}_{i,J} - \bar{X}_{I,j} + \bar{X}_{I,J}$$

This measures deviation from the perfect additive structure.

The **mean squared residue** $H(I, J)$ quantifies bicluster quality:

$$H(I, J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} r^2(X_{ij})$$

Mean Squared Residue (Cheng & Church, 2000)

In practice, biclusters are not perfect. Define the **residue** for entry (i, j):

$$r(X_{ij}) = X_{ij} - \bar{X}_{i,J} - \bar{X}_{I,j} + \bar{X}_{I,J}$$

This measures deviation from the perfect additive structure.

The **mean squared residue** $H(I, J)$ quantifies bicluster quality:

$$H(I, J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} r^2(X_{ij})$$

Definition: A submatrix (I, J) is a δ-**bicluster** if $H(I, J) \leq \delta$ for some threshold $\delta \geq 0$.

Lower $H(I, J)$ indicates stronger coherent (additive) structure. When $H(I, J) = 0$, the bicluster is perfect.

Cheng & Church Algorithm: Finding a Single Bicluster

Greedy approach to find one δ-bicluster with $H(I, J) \leq \delta$:

1. **Node deletion:** Iteratively remove rows and columns with high residue until $H(I, J) \leq \delta$
2. **Node addition:** Add back rows/columns that maintain $H(I, J) \leq \delta$ to maximize bicluster size

This finds a single high-quality bicluster, but what about finding multiple biclusters?

Finding Multiple Biclusters: The Masking Problem

Cheng & Church's approach: After finding a bicluster (I, J):

1. Mask the found bicluster by replacing X_{ij} (for $i \in I, j \in J$) with random noise
2. Run the algorithm again on the masked matrix to find the next bicluster
3. Repeat until desired number of biclusters found

Finding Multiple Biclusters: The Masking Problem

Cheng & Church's approach: After finding a bicluster (I, J):

1. Mask the found bicluster by replacing X_{ij} (for $i \in I, j \in J$) with random noise
2. Run the algorithm again on the masked matrix to find the next bicluster
3. Repeat until desired number of biclusters found

The problem is that masking with random noise can:

- Destroy information about overlapping biclusters
- Prevent discovery of biclusters that share rows/columns
- Introduce artifacts that affect subsequent searches

FLOC: FLexible Overlapped Biclustering

Instead of masking with noise, treat previously found biclusters as "missing" data.

FLOC's approach:

- When computing $H(I, J)$ for a new bicluster candidate, *exclude* entries already assigned to previous biclusters
- This allows rows/columns to participate in multiple biclusters
- But we don't need to add masking noise

FLOC: FLexible Overlapped Biclustering

Instead of masking with noise, treat previously found biclusters as "missing" data.

FLOC's approach:

- When computing $H(I, J)$ for a new bicluster candidate, *exclude* entries already assigned to previous biclusters
- This allows rows/columns to participate in multiple biclusters
- But we don't need to add masking noise

All we need now is a way to handle missing data reliably.

FLOC: Formal Definition

A (δ, ϑ)-bicluster (I, J) must satisfy:

Coherence criterion:

$$H_{\text{available}}(I, J) \leq \delta$$

where $H_{\text{available}}$ is computed omitting missing entries.

FLOC: Formal Definition

A (δ, ϑ)-bicluster (I, J) must satisfy:

Coherence criterion:

$$H_{\text{available}}(I, J) \leq \delta$$

where $H_{\text{available}}$ is computed omitting missing entries.

Occupancy criteria:

- For each row $i \in I$: at least fraction ϑ of columns in J must be not be missing
- For each column $j \in J$: at least fraction ϑ of rows in I must be not be missing

Computing $H_{\text{available}}$: Notation

For a bicluster (I, J), define

Available columns for row i:

$$J_i' = \{j \in J : \text{entry } (i,j) \text{ is available (not in previous biclusters)}\}$$

Computing $H_{\text{available}}$: Notation

For a bicluster (I, J), define

Available columns for row i:

$$J_i' = \{j \in J : \text{entry } (i,j) \text{ is available (not in previous biclusters)}\}$$

Available rows for column j:

$$I_j' = \{i \in I : \text{entry } (i,j) \text{ is available (not in previous biclusters)}\}$$

Computing $H_{\text{available}}$: Notation

For a bicluster (I, J), define

Available columns for row i:

$$J_i' = \{j \in J : \text{entry } (i,j) \text{ is available (not in previous biclusters)}\}$$

Available rows for column j:

$$I_j' = \{i \in I : \text{entry } (i,j) \text{ is available (not in previous biclusters)}\}$$

Available entries set: Let $\mathcal{O}_{IJ} = \{(i,j) : i \in I, j \in J, i \in I_j', j \in J_i'\}$ be all observed (available) entries.

Computing $H_{\text{available}}$: Modified Formulas

Now, we compute means over available entries only:

Row mean for row i:

$$\bar{X}_{i,J}^{\text{avail}} = \frac{1}{|J_i'|} \sum_{j \in J_i'} X_{ij}$$

Column mean for column j:

$$\bar{X}_{I,j}^{\text{avail}} = \frac{1}{|I_j'|} \sum_{i \in I_j'} X_{ij}$$

Overall bicluster mean:

$$\bar{X}_{I,J}^{\text{avail}} = \frac{1}{|\mathcal{O}_{IJ}|} \sum_{(i,j) \in \mathcal{O}_{IJ}} X_{ij}$$

Computing $H_{\text{available}}$: Residue and Quality Measure

Define the **residue using available means:**

$$r_{\text{avail}}(X_{ij}) = X_{ij} - \bar{X}_{i,J}^{\text{avail}} - \bar{X}_{I,j}^{\text{avail}} + \bar{X}_{I,J}^{\text{avail}}$$

This measures how much entry (i, j) deviates from the additive structure based on available data only.

Computing $H_{\text{available}}$: Residue and Quality Measure

Define the **residue using available means:**

$$r_{\text{avail}}(X_{ij}) = X_{ij} - \bar{X}_{i,J}^{\text{avail}} - \bar{X}_{I,j}^{\text{avail}} + \bar{X}_{I,J}^{\text{avail}}$$

This measures how much entry (i, j) deviates from the additive structure based on available data only.

Mean squared residue over observed entries:

$$H_{\text{available}}(I, J) = \frac{1}{|\mathcal{O}_{IJ}|} \sum_{(i,j) \in \mathcal{O}_{IJ}} r_{\text{avail}}^2(X_{ij})$$

Computing $H_{\text{available}}$: Residue and Quality Measure

Define the **residue using available means:**

$$r_{\text{avail}}(X_{ij}) = X_{ij} - \bar{X}_{i,J}^{\text{avail}} - \bar{X}_{I,j}^{\text{avail}} + \bar{X}_{I,J}^{\text{avail}}$$

This measures how much entry (i, j) deviates from the additive structure based on available data only.

Mean squared residue over observed entries:

$$H_{\text{available}}(I, J) = \frac{1}{|\mathcal{O}_{IJ}|} \sum_{(i,j) \in \mathcal{O}_{IJ}} r_{\text{avail}}^2(X_{ij})$$

A bicluster (I, J) is a (δ, ϑ)-bicluster if:

- $H_{\text{available}}(I, J) \leq \delta$ (coherence)
- $|J_i'|/|J| \geq \vartheta$ for all $i \in I$ and $|I_j'|/|I| \geq \vartheta$ for all $j \in J$ (occupancy)

Plaid Model: Motivation

Limitations of previous methods:

- Hartigan: Creates strict partitions (no overlaps)
- Cheng & Church + masking: Destroys overlap information
- FLOC: Allows overlaps but treats them implicitly

Plaid Model: Motivation

Limitations of previous methods:

- Hartigan: Creates strict partitions (no overlaps)
- Cheng & Church + masking: Destroys overlap information
- FLOC: Allows overlaps but treats them implicitly

Real data often has overlapping patterns, i.e. a row/column can belong to multiple biclusters simultaneously with different effects.

Plaid Model: Motivation

Limitations of previous methods:

- Hartigan: Creates strict partitions (no overlaps)
- Cheng & Church + masking: Destroys overlap information
- FLOC: Allows overlaps but treats them implicitly

Real data often has overlapping patterns, i.e. a row/column can belong to multiple biclusters simultaneously with different effects.

For example, say a student is taking a maths test

- Student i on question j gains $+10$ points from strong algebra skills (bicluster 1)
- Student i on question j gains $+5$ points from good test-taking strategies (bicluster 2)
- A strategic student with algebra skills receives *both* boosts: $+15$ points total

Plaid Model: Additive Layer Structure

The **plaid model** (Lazzeroni & Owen, 2002) represents data as a sum of bicluster "layers":

$$X_{ij} = \mu_0 + \sum_{k=1}^{K} \rho_{ik} \kappa_{jk} \theta_{ijk}$$

Plaid Model: Additive Layer Structure

The **plaid model** (Lazzeroni & Owen, 2002) represents data as a sum of bicluster "layers":

$$X_{ij} = \mu_0 + \sum_{k=1}^{K} \rho_{ik} \kappa_{jk} \theta_{ijk}$$

where:

- μ_0 is the background/baseline value (layer 0)
- K is the number of biclusters (layers 1 through K)
- $\rho_{ik} \in \{0, 1\}$ indicates if row i belongs to bicluster k
- $\kappa_{jk} \in \{0, 1\}$ indicates if column j belongs to bicluster k
- θ_{ijk} is the contribution of bicluster k to entry (i, j)

When $\rho_{ik} = 1$ and $\kappa_{jk} = 1$, entry (i, j) receives contribution θ_{ijk} from layer k.

Plaid Model: Understanding Overlaps

For a given entry X_{ij}, the observed value is:

$$X_{ij} = \mu_0 + \sum_{k:\rho_{ik}=1,\kappa_{jk}=1} \theta_{ijk}$$

Different scenarios:

- If (i, j) belongs to no biclusters: $X_{ij} = \mu_0$ (background only)

Plaid Model: Understanding Overlaps

For a given entry X_{ij}, the observed value is:

$$X_{ij} = \mu_0 + \sum_{k:\rho_{ik}=1,\kappa_{jk}=1} \theta_{ijk}$$

Different scenarios:

- If (i, j) belongs to no biclusters: $X_{ij} = \mu_0$ (background only)
- If (i, j) belongs to bicluster k only: $X_{ij} = \mu_0 + \theta_{ijk}$

Plaid Model: Understanding Overlaps

For a given entry X_{ij}, the observed value is:

$$X_{ij} = \mu_0 + \sum_{k:\rho_{ik}=1,\kappa_{jk}=1} \theta_{ijk}$$

Different scenarios:

- If (i,j) belongs to no biclusters: $X_{ij} = \mu_0$ (background only)
- If (i,j) belongs to bicluster k only: $X_{ij} = \mu_0 + \theta_{ijk}$
- If (i,j) belongs to biclusters 1 and 2: $X_{ij} = \mu_0 + \theta_{ij1} + \theta_{ij2}$

Plaid Model: Understanding Overlaps

For a given entry X_{ij}, the observed value is:

$$X_{ij} = \mu_0 + \sum_{k:\rho_{ik}=1,\kappa_{jk}=1} \theta_{ijk}$$

Different scenarios:

- If (i, j) belongs to no biclusters: $X_{ij} = \mu_0$ (background only)
- If (i, j) belongs to bicluster k only: $X_{ij} = \mu_0 + \theta_{ijk}$
- If (i, j) belongs to biclusters 1 and 2: $X_{ij} = \mu_0 + \theta_{ij1} + \theta_{ij2}$

Effects are additive and explicit. Unlike FLOC (which treats overlaps as "missing"), the plaid model estimates how much each bicluster contributes.

Specifying the Layer Effects

The plaid model allows flexibility in how we specify θ_{ijk}, the contribution of layer k to entry (i, j).

Common specifications:

Specifying the Layer Effects

The plaid model allows flexibility in how we specify θ_{ijk}, the contribution of layer k to entry (i, j).

Common specifications:

1. **Constant layer model:**

$$\theta_{ijk} = \theta_k$$

Every entry in bicluster k receives the same contribution.

Specifying the Layer Effects

The plaid model allows flexibility in how we specify θ_{ijk}, the contribution of layer k to entry (i, j).

Common specifications:

1. **Constant layer model:**

$$\theta_{ijk} = \theta_k$$

   Every entry in bicluster k receives the same contribution.

2. **Additive row-column model:**

$$\theta_{ijk} = \theta_k + \alpha_{ik} + \beta_{jk}$$

   Layer k has a baseline θ_k plus row effects α_{ik} and column effects β_{jk}.

Specifying the Layer Effects

The plaid model allows flexibility in how we specify θ_{ijk}, the contribution of layer k to entry (i, j).

Common specifications:

1. **Constant layer model:**

$$\theta_{ijk} = \theta_k$$

   Every entry in bicluster k receives the same contribution.

2. **Additive row-column model:**

$$\theta_{ijk} = \theta_k + \alpha_{ik} + \beta_{jk}$$

   Layer k has a baseline θ_k plus row effects α_{ik} and column effects β_{jk}.

3. **Multiplicative model:**

$$\theta_{ijk} = \theta_k \times \alpha_{ik} \times \beta_{jk}$$

   Useful for modeling proportional effects.

Objective function

To fit the plaid model, we minimize the residual sum of squares (RSS):

$$\text{RSS} = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(X_{ij} - \mu_0 - \sum_{k=1}^{K}\rho_{ik}\kappa_{jk}\theta_{ijk}\right)^2$$

Objective function

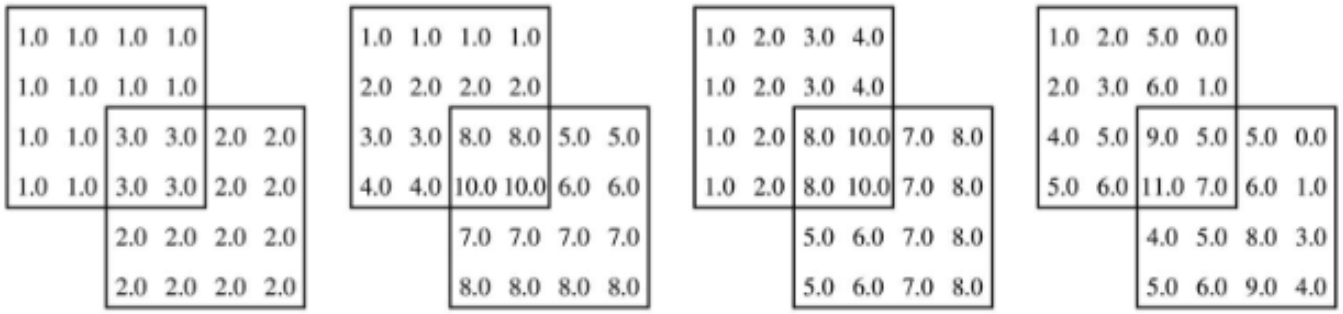
To fit the plaid model, we minimize the residual sum of squares (RSS):

$$\text{RSS} = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(X_{ij} - \mu_0 - \sum_{k=1}^{K}\rho_{ik}\kappa_{jk}\theta_{ijk}\right)^2$$

The optimization is typically performed using an iterative algorithm:

- Alternate between updating bicluster memberships (ρ_{ik}, κ_{jk}) and layer effects (θ_{ijk})
- Continue until convergence (changes in RSS are small)

Bicluster Visualization Example



2. Overlapping biclusters with general additive model. (a) Constant biclusters, (b) constant rows, (c) constant columns, and (d) coherent values.

From Madeira & Oliviera

Bicluster Structure Diagram

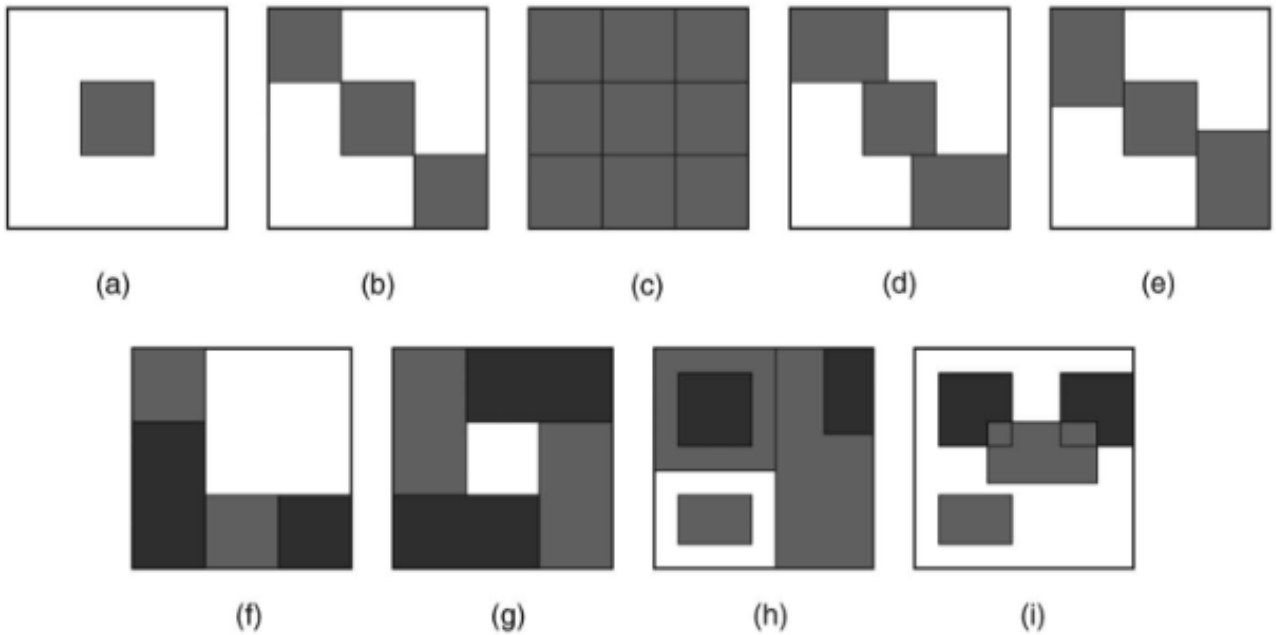


Fig. 4. Bicluster structure. (a) Single bicluster, (b) exclusive row and column biclusters, (c) checkerboard structure, (d) exclusive rows biclusters, (e) exclusive columns biclusters, (f) nonoverlapping biclusters with tree structure, (g) nonoverlapping nonexclusive biclusters, (h) overlapping biclusters with hierarchical structure, and (i) arbitrarily positioned overlapping biclusters.

From Madeira & Oliviera

Summary: Biclustering Methods

1. **Hartigan's method:** Partitions matrix into non-overlapping constant biclusters by minimizing within-cluster variance

Summary: Biclustering Methods

1. **Hartigan's method:** Partitions matrix into non-overlapping constant biclusters by minimizing within-cluster variance
2. **Cheng & Church:** Finds coherent biclusters using mean squared residue $H(I, J)$; limited by masking strategy for multiple biclusters

Summary: Biclustering Methods

1. **Hartigan's method:** Partitions matrix into non-overlapping constant biclusters by minimizing within-cluster variance
2. **Cheng & Church:** Finds coherent biclusters using mean squared residue $H(I, J)$; limited by masking strategy for multiple biclusters
3. **FLOC:** Extends Cheng & Church by treating found biclusters as missing data, enabling natural overlaps

Summary: Biclustering Methods

1. **Hartigan's method:** Partitions matrix into non-overlapping constant biclusters by minimizing within-cluster variance
2. **Cheng & Church:** Finds coherent biclusters using mean squared residue $H(I, J)$; limited by masking strategy for multiple biclusters
3. **FLOC:** Extends Cheng & Church by treating found biclusters as missing data, enabling natural overlaps
4. **Plaid model:** Explicitly models overlapping biclusters as additive layers with flexible effect specifications