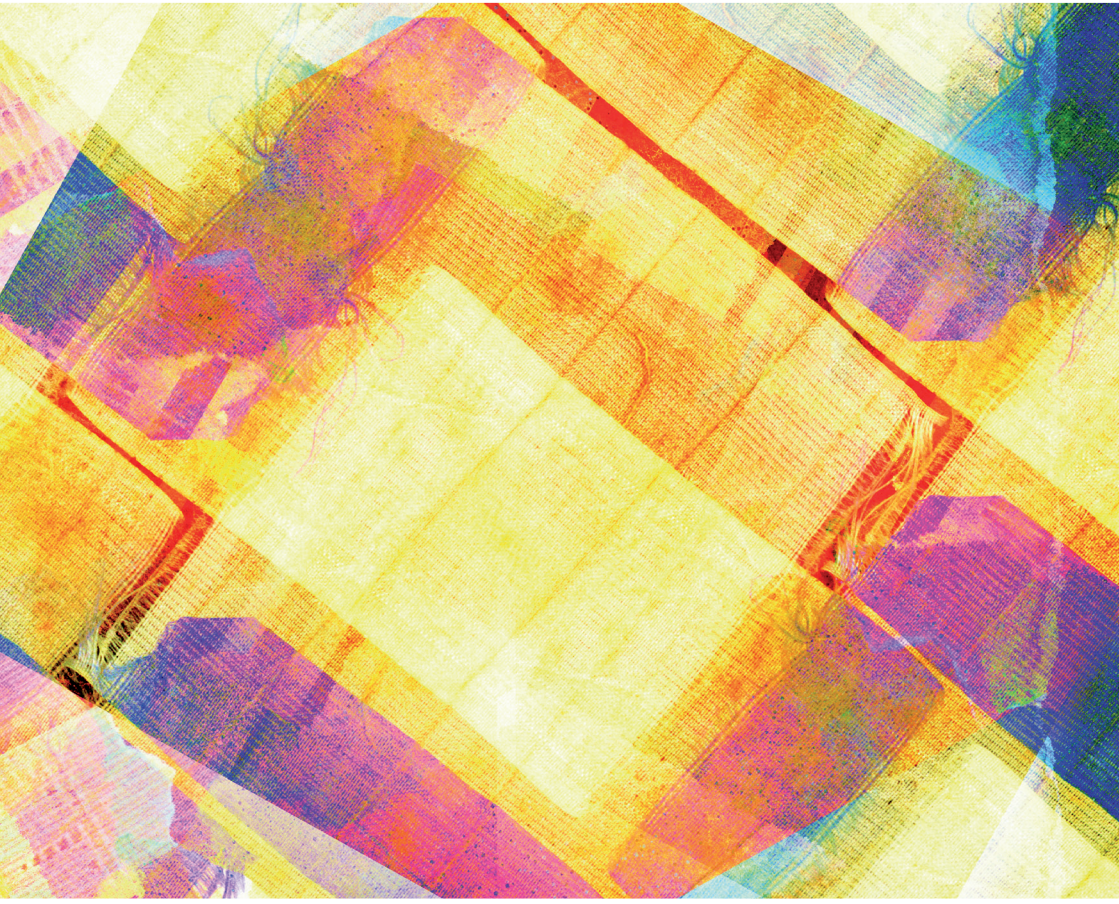
Documentos

ISSN 1677-9274
Dezembro, 2010
109

Estudo de algoritmos de *biclustering* para a análise de expressão gênica utilizando a tecnologia de microarranjo



Embrapa

ISSN 1677-9274
Dezembro, 2010

Empresa Brasileira de Pesquisa Agropecuária
Embrapa Informática Agropecuária
Ministério da Agricultura, Pecuária e Abastecimento

Documentos 109

Estudo de algoritmos de *biclustering* para a análise de expressão gênica utilizando a tecnologia de microarranjo

Roberto Hiroshi Higa
Luciana Correia de Almeida Regitano
Adriana Mércia Guaratini Ibelli
Isabel Luzia Nori dos Santos

Embrapa Informática Agropecuária
Campinas, SP
2010

Embrapa Informática Agropecuária
Av. André Tosello, 209 - Barão Geraldo
Caixa Postal 6041 - 13083-886 - Campinas, SP
Fone: (19) 3211-5700 - Fax: (19) 3211-5754
www.cnptia.embrapa.br
sac@cnptia.embrapa.br

Comitê de Publicações

Presidente: *Silvia Maria Fonseca Silveira Massruhá*

Membros: *Poliana Fernanda Giachetto, Roberto Hiroshi Higa, Stanley Robson de Medeiros Oliveira, Maria Goretti Gurgel Praxedes, Adriana Farah Gonzalez, Neide Makiko Furukawa*

Membros suplentes: *Alexandre de Castro, Fernando Attique Máximo, Paula Regina Kuser Falcão*

Supervisor editorial: *Neide Makiko Furukawa*

Revisor de texto: *Adriana Farah Gonzalez*

Normalização bibliográfica: *Maria Goretti Gurgel Praxedes*

Editoração eletrônica/Arte capa: *Suzilei Almeida Carneiro*

Fotos da capa: *Imagens livres disponíveis em <http://www.stock.schng>*

Secretária: *Carla Cristiane Osawa*

1ª edição on-line 2010

Todos os direitos reservados.
A reprodução não autorizada desta publicação, no todo ou em parte, constitui violação dos direitos autorais (Lei no 9.610).

Dados Internacionais de Catalogação na Publicação (CIP)
Embrapa Informática Agropecuária

Higa, Roberto Hiroshi.

Estudo de algoritmos de biclustering para a análise de expressão gênica utilizando a tecnologia de microarranjo / Roberto Hiroshi Higa... [et al.]. - Campinas : Embrapa Informática Agropecuária, 2010.

43 p. : il. – (Documentos / Embrapa Informática Agropecuária ; ISSN 1677-9274, 109).

1. Análise *biclustering*. 2. Expressão gênica 3. Algoritmos . I. Higa, Roberto Hiroshi. II. Título. III. Série.

570.285 CDD (21. ed.)

© Embrapa 2010

Autor

Roberto Hiroshi Higa
Doutor em Engenharia Elétrica
Pesquisador da Embrapa Informática Agropecuária
Av. André Tosello, 209, Barão Geraldo
Caixa Postal 6041 - 13083-970 - Campinas, SP
Telefone: (19) 3211-5862
e-mail: roberto@cnptia.embrapa.br

Luciana Correia de Almeida Regitano
Doutora em Genética e Melhoramento
Pesquisador da Embrapa Pecuária Sudeste
Rodovia Washington Luiz, km 234
13560-970 - São Carlos, SP
Telefone: (16) 3411-5600
e-mail: luciana@cppse.embrapa.br

Adriana Mércia Guaratini Ibelli
Bolsista CAPES / Embrapa Pecuária Sudeste
e-mail: adriana.ibelli@gmail.com

Isabel Luzia Nori dos Santos
Estagiária da Embrapa Informática Agropecuária
e-mail: isabel.luzia@gmail.com

Apresentação

Um grande número de abordagens de agrupamentos foi sugerido para análise de dados de expressão gênica, obtidos por meio de experimentos de microarranjo. No entanto, os resultados da aplicação dos algoritmos mais tradicionais, para agrupamento de genes, apresentam algumas limitações, que decorrem da existência de grupos dependentes dos efeitos das condições experimentais em que a atividade dos genes é não correlacionada. Uma limitação semelhante existe quando o agrupamento de condições é realizado. Por isso, uma série de algoritmos tem sido proposta para encontrar, simultaneamente, agrupamentos de genes e amostras (que correspondem às condições). Esse tipo de agrupamento, normalmente, referenciado como biclustering, tem por objetivo encontrar submatrizes da matriz dados, ou seja, subgrupos de genes e subgrupos de condições em que os genes apresentam expressão altamente correlacionada em todo o subgrupo de condições.

O objetivo deste trabalho é relatar e resumir um estudo realizado em relação: (i) aos conceitos referentes à análise de biclustering; (ii) a um conjunto de algoritmos disponíveis em pacotes do software estatístico R (R DEVELOPMENT CORE TEAM, 2010); e, (iii) a um conjunto de exemplos de utilização desses algoritmos em dados reais de expressão gênica utilizando microarranjos (IBELLI et al., 2010).

Como esses algoritmos têm sido utilizados não só na análise de expressão gênica, mas também em áreas como a recuperação de informação e mineração de dados, este estudo pode ser potencialmente útil a pesquisadores desas outras áreas.

Kleber Xavier Sampaio de Souza
Chefe Geral
Embrapa Informática Agropecuária

Sumário

Estudo de algoritmos de *biclustering* para a análise de expressão gênica utilizando a tecnologia de microarranjo

Roberto Hiroshi Higa
Luciana Correia de Almeida Regitano
Adriana Mércia Guaratini Ibelli
Isabel Luzia Nori dos Santos

Introdução

A tecnologia de microarranjo permite que as expressões gênicas de um grande número de genes sejam simultaneamente avaliadas em diferentes condições, que formam amostras observadas sobre elas. As condições podem corresponder a diferentes pontos no tempo, tecidos, condições experimentais ou condições ambientais. Um passo crucial na análise desse tipo de dado é a extração de informações biologicamente relevantes.

Em geral, dados de expressão gênica são organizados em uma matriz, onde cada linha corresponde a um gene e cada coluna a uma condição. Cada elemento dessa matriz contém um número real que reflete o logaritmo da abundância relativa de mRNA de um determinado gene sob uma condição específica.

Alguns dos objetivos da análise de dados de expressão gênica são:

1. Agrupar genes de acordo com sua expressão sob múltiplas condições;
2. Classificar um novo gene, dada a sua expressão e a expressão de outros genes com classificação conhecida;
3. Agrupar as condições com base nas expressões de um número determinado de genes;

4. Classificar uma nova amostra, dada a expressão de um conjunto de genes e de outras amostras com a classificação conhecida.

Técnicas de agrupamento podem ser utilizadas para agrupar tanto genes quanto condições e, portanto, para atingir os objetivos 1 e 3 e, indiretamente, os objetivos 2 e 4.

Entretanto, a aplicação de técnicas de agrupamento a dados de expressão gênica esbarram em uma dificuldade: muitos padrões de expressão são comuns a um grupo de genes apenas sob condições experimentais específicas. Descobrir tais padrões de expressão local pode ser importante para revelar a presença de vias de regulação gênica que, de outra maneira, não são aparentes. Por essa razão, esforços têm sido envidados no sentido de desenvolver algoritmos que vão além do paradigma de agrupamento, que possam descobrir padrões locais em dados de microarranjos. Revisões sobre o assunto podem ser encontradas em (MADEIRA et al., 2004).

Quando algoritmos de agrupamento são utilizados, todas as condições são utilizadas na definição dos grupos de genes ,ou seja, obtém-se um modelo global dos agrupamentos dos dados. Analogamente, todos os genes são utilizados na definição de grupos de condições, enquanto em algoritmos de *biclustering*, os genes são agrupados utilizando apenas um subconjunto das condições e as condições agrupadas utilizando um subconjunto dos genes, por isso, diz-se que os métodos de *biclustering* produzem modelos locais.

1. Caracterização

Segundo Madeira et al. (2004), algoritmos de *biclustering* podem ser analisados sob diferentes critérios. Dentre estes, um critério interessante considera o tipo de *bicluster* que o algoritmo se propõe a encontrar. Quatro tipos de *biclusters* são definidos:

- *Biclusters* com valores constantes: *bicluster* constituídos por subconjuntos de linhas e subconjuntos de colunas com valores constantes são o alvo dos algoritmos mais simples; Um exemplo desse tipo de *bicluster* é apresentado na Figura 1 (a);
- *Biclusters* com valores constantes em linhas ou colunas: outras

abordagens identificam subconjuntos de linhas e subconjuntos de colunas com valores constantes nas linhas ou nas colunas da matriz de dados; o *bicluster* apresentado na Figura 1 (b) é um exemplo de *bicluster* com linhas constantes, enquanto o *bicluster* apresentado na Figura 1(c) é um exemplo de *bicluster* com colunas constantes;

- *Biclusters* com valores coerentes: por valores coerentes entende-se *biclusters* onde cada linha ou coluna pode ser obtida a partir das outras pela adição ou multiplicação por uma constante; os *biclusters* das Figuras 1 (d) e 1 (e) são exemplos deste tipo de *bicluster*;
- *Biclusters* com evoluções coerentes: neste caso, considera-se que os valores dos elementos da matriz de dados são simbólicos e entende-se por evoluções coerentes *biclusters* com subconjuntos de linhas e/ou colunas com comportamento coerente, independente dos valores numéricos exatos; a propriedade de coevolução pode ser observada em todo o *bicluster*, ou seja, tanto nas linhas quanto nas colunas do *bicluster* apresentado na Figura 1 (f), nas suas linhas apresentada na Figura 1 (g) ou nas suas colunas como mostra a Figura 1 (h).

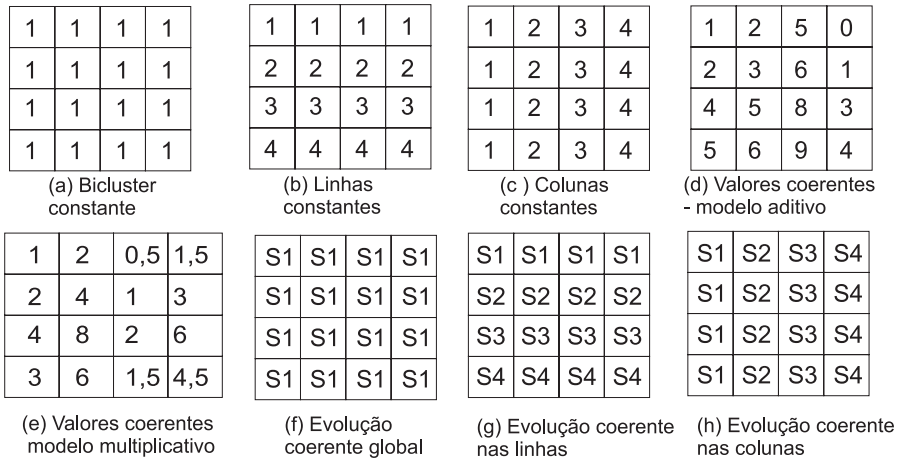


Figura 1. Tipos de bi*clusters*.
Fonte: Adaptado de Madeira et al. (2004)

Uma forma de se identificar visualmente *biclusters* em uma matriz de dados A consiste em construir uma imagem colorida de A, denominada *heatmap*, na qual cada elemento da matriz é colorido, de acordo com o valor de a_{ij}. Nesse caso, a ideia é reordenar as linhas e colunas similares de A, de forma a evidenciar blocos similares e aplicar-lhes a mesma cor.

As estruturas de *biclusters* consideradas por diferentes algoritmos apresentam restrições variadas. Alguns assumem que, para propósitos visuais, todos os *biclusters* encontrados podem ser observados diretamente na matriz de dados e exibidos como uma representação contígua, depois da execução de um processo de reordenação de linhas e colunas. Outras assumem que os *biclusters* são completos, isto é, cada linha e cada coluna da matriz de dados pertence a, pelo menos, um *bicluster*. Por fim, alguns métodos assumem uma estrutura de *biclusters* bem menos restrita e mais próxima do cenário apresentado por dados reais, em que algumas linhas ou colunas podem não pertencer a nenhum *bicluster* e que pode existir sobreposição entre *biclusters*. A Figura 2 ilustra essas diferentes estruturas de *biclusters*.

2. Algoritmos

Considere uma matriz de dados A_{nxm} com elementos aij reais, onde cada elemento a_{ij} representa o nível do gene i sob a condição j. $X = \{x_1, \ldots, x_n\}$ é o conjunto de linhas de A e $Y = \{y_1, \ldots, y_m\}$ é o conjunto de colunas de A. Então, A pode ser denotada como A = (X,Y). Além disso, se $I \subseteq X$ e $J \subseteq Y$ e denotam subconjuntos de linhas e colunas de A, então, AIJ = (I,J) denota uma submatriz de A formada pelas linhas I e colunas J.

A partir dessa notação, um subconjunto de linhas com expressão semelhante é denotado por A_{IY} = (I,Y), onde $I = \{x_1, \ldots, x_k\}$ é um subconjunto de linhas, $I \subseteq X$ e $k \leq n$. Analogamente, um subconjunto de colunas com expressão semelhante é denotado por A_{XJ} = (X,J), onde $J = \{y_1, \ldots, y_{ks}\}$ é um subconjunto de colunas, $I \subseteq X$ e $s \leq m$. Um *bicluster* (I,J) é uma submatriz, da matriz de dados A.

Além disso, considerando a mesma notação, a média da i-ésima linha de um *bicluster*, a_{iJ}, é dada por $a_{iJ} = \frac{1}{|J|}\sum_{j \in J} a_{ij}$ a média da j-ésima coluna de

um *bicluster*, a_{ij}, é dada por $a_{Ij} = \frac{1}{|I|}\sum_{i \in I} a_{ij}$.

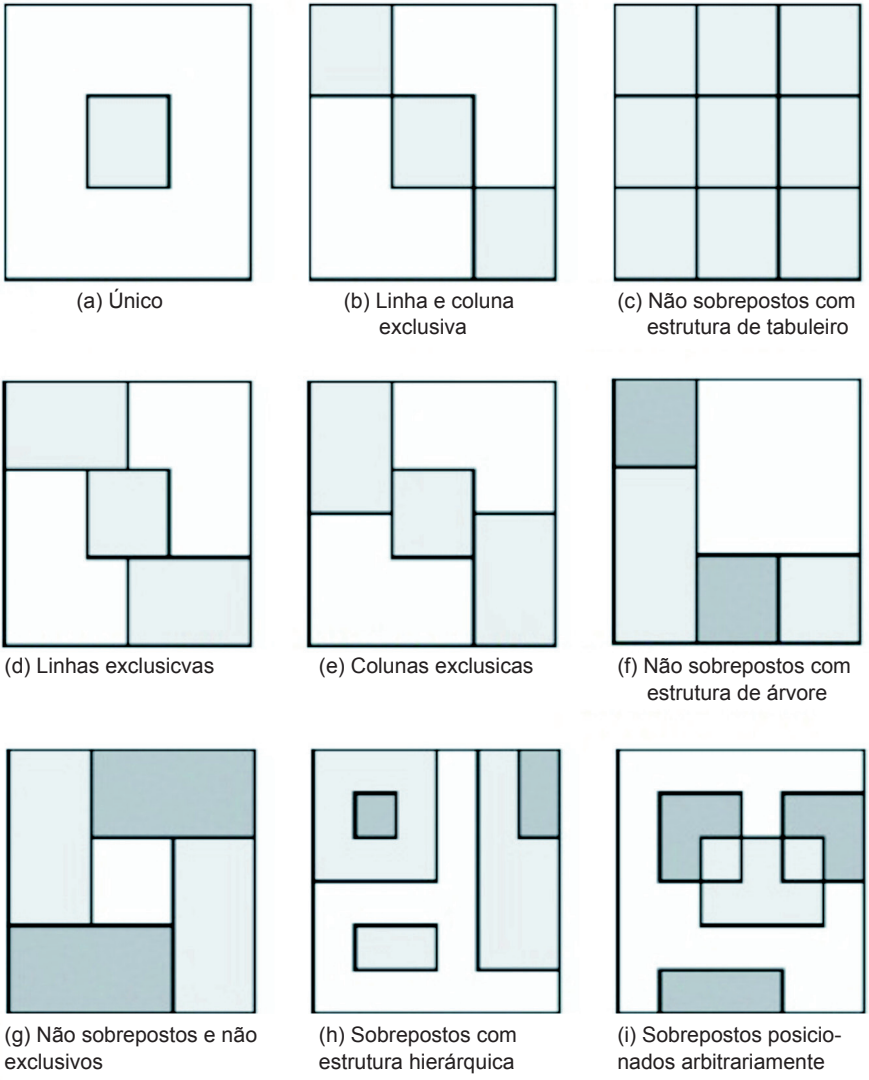


Figura 2. Estrutura de *biclusters*.
Fonte: Adaptado de Madeira et al. (2004).

Por fim, a média entre todos os elementos de um *bicluster*, a_{IJ}, é dada por $a_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} a_{ij} = \frac{1}{|I|} \sum_{i \in I} a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{Ij}$.

Quando o objetivo de um algoritmo de *biclustering* é encontrar um ou mais *biclusters* constantes, é natural considerar maneiras de reordenar as linhas e as colunas da matriz de dados para agrupar linhas e colunas semelhantes e descobrir *biclusters* com valores similares. Quando dados de expressão gênica são utilizados, *biclusters* com valores constantes revelam subconjuntos de genes com valores de expressão similares dentro de um subconjunto de condições. Um *bicluster* com valores constantes perfeito é uma submatriz de A=(I,J), onde todos os valores dentro do *bicluster* são iguais, ou seja, para todo $i \in I$ e todo $j \in J$: $a_{ij} = \mu$. Embora esses *biclusters* ideais possam ser encontrados em algumas matrizes de dados, em dados reais, *biclusters* com valores constantes encontram-se mascarados por ruídos. Em geral, considera-se que $a_{ij} = \mu + \eta_{ij}$ e *biclusters* com valores constantes são avaliados utilizando alguma métrica baseada na variância dos dados pertencentes ao *bicluster*.

Um *bicluster* perfeito, com valores constantes nas linhas, é uma submatriz (I,J), onde todos os valores dentro do *bicluster* podem ser obtidos usando uma das seguintes expressões:

$$a_{ij} = \mu + \alpha_i \quad (1)$$

$$a_{ij} = \mu \times \alpha_i \quad (2)$$

onde μ é o valor típico dentro do *bicluster* e αi é o ajuste para a linha , podendo ser aditivo (1) ou multiplicativo (2).

Analogamente, um *bicluster*, com valores constantes nas colunas, é uma submatriz (I,J), onde todos os valores dentro do *bicluster* podem ser obtidos usando uma das seguintes expressões:

$$a_{ij} = \mu + \beta_j \quad (3)$$

$$a_{ij} = \mu \times \beta_j \quad (4)$$

onde μ é o valor típico dentro do *bicluster* e β_j é o ajuste para a coluna , podendo ser aditivo (3) ou multiplicativo (4).

Esse tipo de *bicluster* não pode ser encontrado simplesmente computando a variância dos valores pertencentes ao *bicluster* ou computando similaridades dentro das linhas e colunas da matriz de dados, como no caso de

biclusters, com valores constantes. A abordagem mais direta para identificar *biclusters* não-constantes é normalizar as linhas ou colunas da matriz de dados usando a média da linha e a média da coluna, respectivamente. Fazendo isso, os *biclusters* com linhas constantes e colunas constantes são transformados em *biclusters* constantes e algoritmos que buscam por *biclusters* com valores contantes podem ser utilizados.

Contudo, por vezes, o que se deseja é identificar *biclusters* mais complexos, que apresentam valores coerentes tanto nas linhas quanto nas colunas. Esse tipo de *bicluster* não pode ser encontrado simplesmente considerando os valores pertencentes ao *bicluster*. Há a necessidade de utilizar-se modelos mais sofisticados que fazem uma análise de variância entre os grupos e usam uma forma particular de covariância entre linhas e colunas no *bicluster* para avaliar a qualidade do *bicluster* ou conjunto de *biclusters* resultantes.

Algoritmos de *biclustering* que procuram identificar *biclusters* com valores coerentes, tanto nas linhas quanto nas colunas da matriz de dados, em geral, baseiam-se em um modelo aditivo. Um *bicluster* perfeito, com valores coerentes (I,J), é definido por um subconjunto de linhas e um subconjunto de colunas, cujos valores obedecem à seguinte relação:

$$a_{ij} = \mu + \alpha_i + \beta_j \qquad (5)$$

onde μ é o valor típico dentro do *bicluster* αi é o ajuste para a linha $i \in I$ e β_j é o ajuste para a coluna $j \in J$. Note que as equações (1) e (3) são casos especiais da equação (5), quando $\alpha_i = 0$ e $\beta_j = 0$, respectivamente. Também é possível utilizar um modelo multiplicativo para modelar os *biclusters*:

$$a_{ij} = \mu' \times \alpha_i' \times \beta\varphi' \qquad (6)$$

Note que as equações (5) e (6) são equivalentes se ,$a_{ij} = \log (a_{ij}')$,

$\mu = \log(\mu')$, $\alpha_i = \log(\alpha_i')$ e $\beta_j = \log(\beta_j')$.

Finalmente, existem ainda algoritmos que procuram identificar evoluções coerentes nas linhas e colunas da matriz de dados, independente dos valores exatos a_{ij}. Por exemplo, no caso de expressão gênica, o objetivo pode ser a identificação de evidências de que um subconjunto de genes é co-regulado sob um conjunto de condições sem considerar o valor real da expressão na matriz de dados.

2.1 Método de Cheng e Church (CC)

Cheng e Church (2000) consideraram um *bicluster* como um subconjunto de linhas e um subconjunto de colunas com um alto índice de similaridade. Sendo a matriz de dados A = (X,Y), eles consideraram que uma submatriz de A, (I,J), é um δ-*bicluster* se seu resíduo quadrático médio, H(I,J) (vide equação 10), é inferior a δ, com $\delta \geq 0$. O objetivo do algoritmo é encontrar os maiores *biclusters* com H inferior a δ.

No caso ideal, quando um δ-*bicluster* é perfeito, todas linhas e colunas apresentam um desvio absolutamente consistente ($\delta = 0$). Isso significa que cada linha ou coluna pode ser gerada pela troca dos valores de outras linhas ou colunas por um valor comum. Quando esse é o caso, $\delta = 0$, cada elemento a_{ij} pode ser definido unicamente pela média da linha, a_{iJ} , pela média da coluna, a_{Ij} , e pela média do *bicluster* a_{IJ} . A diferença $a_{ij} - a_{IJ}$ é o desvio relativo mantido pela coluna j com relação às outras colunas no δ-*bicluster*. O mesmo raciocínio aplicado às linhas leva à definição de que em um δ-*bicluster* perfeito, o valor de um elemento a_{ij} é dado por:

$$a_{iJ} = a_{ij} + a_{Ij} - a_{IJ} \qquad (7)$$

Contudo, para dados reais, em que têm-se δ-*bicluster* não perfeitos, o conceito de resíduo precisa ser introduzido para quantificar a diferença entre o valor observado de um elemento a_{ij} e seu valor esperado pelo modelo (7). O resíduo de um elemento a_{ij} no *bicluster* (I, J) é definido como:

$$r(a_{ij}) = a_{iJ} - a_{ij} - a_{Ij} + a_{IJ} \qquad (8)$$

Assim, o valor de a_{ij} em um *bicluster* não perfeito é dado por:

$$a_{ij} = r(a_{iJ}) + a_{ij} + a_{Ij} - a_{IJ} \qquad (9)$$

onde o valor do resíduo é um indicador da coerência de um valor com relação aos outros valores do *bicluster*, dados os desvios das linhas e colunas relevantes. Quanto menor o resíduo, mais forte é a coerência.

Para avaliar a qualidade global de um δ-*bicluster*, Cheng e Church (2000) definiram o resíduo quadrático médio, H, de um *bicluster* (I, J), como:

$$H = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \qquad (10)$$

O resíduo quadrático médio definido por Cheng e Church assume que não

há valores faltantes na matriz de dados. Para garantir isso, eles substituem os valores faltantes por números aleatórios. O algoritmo pode ser descrito da seguinte forma:

1. **Entrada**: uma matriz de números reais, A, e $\delta \geq 0$, o valor máximo admissível para o resíduo quadrático médio.
2. **Inicialização**: I e J são os conjuntos de linhas e colunas iniciais e $A_{IJ} = A$.
3. **Iteração**: Compute o escore H para cada adição/remoção de linha/coluna possível e escolha a ação que reduza o valor de H. Se não é possível reduzir o valor de H, ou se $H \leq \delta$, retorne A_{IJ}.
4. **Saída**: A_{IJ}, um δ-*bicluster* que é uma submatriz de A definida por um conjunto de linhas I e um conjunto de colunas J, com $H(I, J) \leq \delta$.

2.2 Algoritmo FLOC

Yang et al. (2005) generalizaram a definição de um δ-*bicluster* para tratar valores faltantes isso é, valores para os quais não há observações no conjunto de dados, e evitar a interferência causada pela inclusão de números aleatórios por Cheng e Church. Um δ-*bicluster* é definido por um subconjunto de linhas e um subconjunto de colunas com valores coerentes nas caselas da matriz de dados especificadas (não faltantes). O algoritmo FLOC (*Flexible Overlapped biClustering*) introduz os conceitos de ocupação e volume, utilizados para definir um *bicluster*.

Sendo uma matriz de dados (X, Y) e um limite de ocupação α, um *bicluster* (de ocupação α), pode ser representado por um par (I, J), onde $I \subseteq X$ é um subconjunto de genes e $J \subseteq Y$ é um subconjunto de condições. Para cada gene $i \in I$, $\frac{|Ji'|}{|J|} > \alpha$, onde |Ji'| e |J| são o número de condições especificadas para o gene i no *bicluster* e o número de condições no *bicluster*, respectivamente. De maneira semelhante, para cada condição $j \in J$, $\frac{|Ij'|}{|I|} > \alpha$, onde |Ij'| e |I| são o número de genes especificados sob a condi

ção j no *bicluster* e o número de genes no *bicluster*, respectivamente.

O volume de um *bicluster* (I, J), ν_{IJ} , é definido como o número de valores especificados a_{ij}, tais que i $\in$ I e j $\in$ J. No caso em que todos os valores são especificados, ν_{IJ} = |I| × |J|, onde |I| representa o número de genes e |J| representa o número de condições presentes no *bicluster*.

Sendo um *bicluster* (I, J), a base de um gene x_i é definida como o valor médio de x_i considerando todas as condições especificadas em J:

$$a_{iJ} = \frac{1}{|J'_i|} \sum_{j \in J'_i} a_{ij} \ .$$

Similarmente, a base de uma condição y_j é a definida como o valor médio de y_j considerando todos os genes em I:

$$a_{Ij} = \frac{1}{|I'_j|} \sum_{i \in I'_j} a_{ij}$$

e a base de um *bicluster* é definida com o valor médio de a_{ij} considerando todos os valores especificados na submatriz:

$$a_{Ij} = \frac{1}{\nu_{IJ}} \sum_{i \in I'_j, j \in J'_j} a_{ij} \ .$$

O resíduo de uma casela, a_{ij}, em um *bicluster* é $r(a_{ij}) = a_{iJ} - a_{ij} - a_{Ij} + a_{IJ}$, se a_{ij} é especificado. Em caso contrário, $r(a_{ij}) = 0$. Assim, o resíduo quadrático médio de um *bicluster* (I,J) é dado por:

$$H(I,J) = \frac{1}{\nu_{IJ}} \sum_{i \in I'_j, j \in J'_i} r(a_{ij})^2$$

A descrição do algoritmo FLOC é semelhante ao do algoritmo CC e visa encontrar *biclusters* com resíduo quadrático médio pequenos.

2.3 O Método ISA

O método *Iteractive Signature Algorithm* (ISA) (BERGMANN et al., 2003) considera uma matriz de dados A com elementos a_{cg} reais. A pode ser vista como uma coleção de n vetores linha $A = \begin{pmatrix} \hat{g}^T_1 \\ \hat{g}^T_2 \\ \vdots \\ \hat{g}^T_n \end{pmatrix}$ ou como uma coleção de

m vetores coluna A = (c_1, c_2, …,c_m) e então define duas matrizes de expressão normalizadas como:

$$AG = \begin{pmatrix} \hat{g}^T_1 \\ \hat{g}^T_2 \\ \vdots \\ \hat{g}^T_n \end{pmatrix} \text{ e } AC = (\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_m),$$

onde

e

$$\hat{g}_c = \frac{g_c - \breve{g}_c}{(g_c - \breve{g}_c)} \quad \text{e} \quad \hat{c}_g = \frac{c_g - \breve{c}_g}{|c_g - \breve{c}_g|} \qquad (11)$$

com a média dos vetores g_c dada por $\breve{g}_c$ e a média dos vetores c_g dada por $\breve{c}_g$. Note que os vetores $\hat{g}_c$ e $\hat{c}_g$ têm média zero e comprimento 1. O objetivo é encontrar um conjunto de linhas correlacionadas, $G_p \subseteq G$, junto com um conjunto de colunas, $C_p \subseteq C$. Bergmann et al., (2003) denominam esse conjunto combinado, $M_p = \{G_p, C_p\}$, de Transcription module (TM).

Matematicamente, um TM pode ser definido como:

$$\exists \left(T_C, T_G\right) \textit{ tal que } C_p\left(G_p\right) = c \in C : \tilde{A}^{cg}{}_G > T_C \textit{ e } G_p\left(C_p\right) = g \in G : \tilde{A}^{cg}{}_G > T_G \qquad (12)$$

onde T_C e T_G são dois parâmetros limitantes. A definição acima declara que, para cada condição c no T_M, o nível de expressão médio dos genes no TM é maior que um certo T_C .Reciprocamente, para cada gene g no TM, o nível de expressão médio das condições do TM é maior que um certo TG.

Para reformular e generalizar a definição de um TM em (12), Bergmann et al. (2000) introduziram uma notação vetorial. As linhas de um T_M são representadas por um vetor $g_p = (g_p^{(1)}, g_p^{(2)}, .., g_p^{(n)})^T$ e as colunas por um vetor $c_p = (c_p^{(1)}, c_p^{(2)}, \ldots, c_p^{(m)})^T$ e as seguintes transformações lineares são definidas:

$$c_p^{proj} \equiv A_G.g_p = \begin{matrix} \hat{g}^T_1 g_p \\ \hat{g}^T_2 g_p \\ \vdots \\ \hat{g}^T_n g_p \end{matrix} \quad \text{e } g_p^{proj} \equiv A_C{}^T.c_p = \begin{matrix} \hat{c}^T_1 c_p \\ \hat{c}^T_2 c_p \\ \vdots \\ \hat{c}^T_m c_p \end{matrix} .$$

Dessa forma, a definição de um TM dada em (12) pode ser reescrita como

$$\exists \left(T_C, T_G\right) \textit{ tal que } c_p = f_{tC}\left(c^{proj}{}_p\right) \textit{ e } g_p = f_{tG}\left(g^{proj}{}_p\right) \qquad (13)$$

onde t_C e t_G são valores limites para identificação de linhas e colunas pertencentes a um *bicluster* e f_t é uma função limite. A definição de um TM em (13) pode ser entendida como a aplicação da função f_{tC} a c_p^{proj}, obtendo-se uma componente não-negativa $c_p^{(c)}$ do vetor de condições c_p do TM, se o perfil da linha correspondente, g_c, estiver suficientemente alinhado com o vetor g_p, do TM.

A definição rigorosa de um TM, em princípio, permite determinar os módulos codificados na matriz de expressão, testando todos os possíveis conjuntos {Gp , Cp } para a sua conformidade com a equação (13). No entanto, esta abordagem é computacionalmente inviável. Para contornar essa restrição, Bergmann et al. (2000) propõe uma abordagem iterativa. Sejam:

$$c^{(h+1)} = f_{tC}\left(A_G . g^{(h)}\right) \qquad (14)$$

$$g^{(h+1)} = f_{tG}\left(A_C . c^{(h+1)}\right) \qquad (15)$$

A primeira equação atualiza um vetor de condições $c^{(h+1)}$ a partir de um determinado vetor de genes $g^{(h)}$.A componente $cc^{(h+1)}$ desse vetor é referenciada como escore de condição. Esse escore é não nulo apenas se o perfil gênico correspondente, $\hat{g}_c$, definido em (11), está suficientemente alinhado com o vetor de genes $g_p^{(h)}$. Na etapa seguinte, definida por (15), o componente (ou escore de gene) $g_g^{(h+1)}$ do vetor de genes $gp^{(h)}$ é atualizado com um valor não nulo se o perfil de condições correspondentes, $\hat{c}_g$ está suficientemente alinhado com o vetor de condições $cp^{(h+1)}$. Em geral, o vetor inicial é escolhido aleatoriamente. A série $\{g^{(0)}, g^{(1)}, g^{(3)}, \ldots.\}$ converge rapidamente, tal que um ponto fixo é definido como um vetor que genes, $g^{(h^*)}$, que satisfaz à seguinte condição (BERGMANN et al., 2000):

$$\frac{|g^{(fix)} - g^h|}{|g^{(fix)} + g^h|} < \varepsilon$$

para todo h maior que um certo número de iterações. O parâmetro ε determina a precisão do ponto fixo, $g^{(fix)}$. $g^{(fix)}$ depende tanto da semente $g^{(0)}$ quanto dos limites t_G e t_C, que são parâmetros fixos. Assim, uma TM é definida pelo par $(g^{(fix)}), c^{(fix)})$ que resolve a equação (13) e determinados utilizando-se as equações (14) e (15).

2.4 Algoritmo Bimax

O modelo considerado pelo algoritmo Bimax (PRELIC et al., 2006) pressupõe dois possíveis níveis de expressão por gene: sem alterações e com alterações em relação a uma experiência controlada. Assim, um conjunto de m experimentos de microarranjo para n genes pode ser representado por uma matriz binária E(n×m), onde a célula eij é 1 quando o gene i responde à condição j, caso contrário seu valor é 0. Um *bicluster* (G, C) corresponde a um subconjunto de genes $G \subseteq 1, \ldots, n$, que respondem conjuntamente a um subconjunto de amostras $C \subseteq 1, \ldots, m$. Em outras palavras, o par (G, C) define uma submatriz de E em que todos os elementos são iguais a 1.

O algoritmo Bimax utiliza uma estratégia de dividir-e-conquistar para identificar áreas que contém apenas zeros e excluí-las de inspeções futuras (Figura 3).

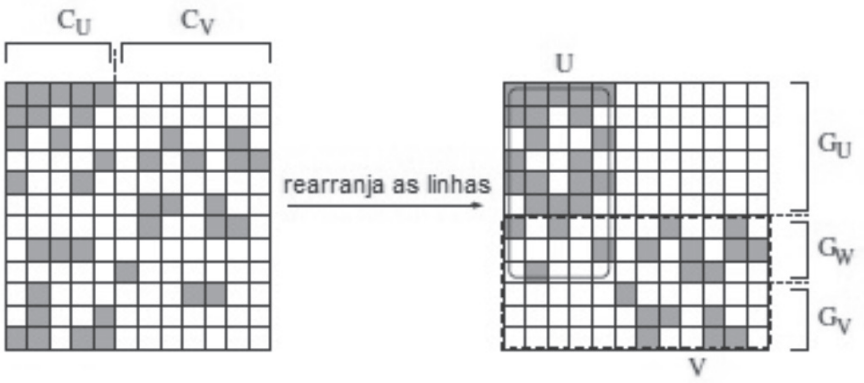


Figura 3. Ilustração do algoritmo Bimax.
Fonte: Adaptado de Prelic et al. (2006).

Mais especificamente, a matriz E é particionada em 3 submatrizes, uma das quais contém apenas 0's e, portanto, pode ser desconsiderada nos passo seguintes. O algoritmo, então, é aplicado recursivamente às duas submatrizes restantes. A recursão termina quando representa um *bicluster*, ou seja, contém apenas 1's.

2.5 Algoritmo *xMotifs*

Murali e Kasif (2003) definem um *bicluster*, denominado *xMotif*, como um subconjunto de genes (linhas) que é simultaneamente conservado em um subconjunto de condições (colunas), sendo que o nível de expressão de um gene é considerado conservado em um subconjunto de condições se o gene está em um mesmo estado em cada uma das condições do subconjunto. Mais precisamente, dado um conjunto de genes cujos níveis de expressão foram medidos em diferentes condições e os parâmetros $0 < \alpha, \beta < 1$ definidos pelo usuário, um *xMotif* é definido como um par (C, G), onde C é um subconjunto das amostras e G é um subconjunto de genes que satisfaz às seguintes condições:

- **Tamanho:** o número de amostras em C é pelo menos uma fração α de todas as condições,
- **Conservação:** cada gene em G é conservado em todas as amostras em C, ou seja, o gene está no mesmo estado em todas as amostras em C, e
- **Maximização**: para cada gene que não está em G, o gene é conservado em no máximo uma fração β das amostras de C.

Essa definição suporta a ocorrência de muitos *xMotifs* em um conjunto de dados de expressão gênica. Entre todos esses, o algoritmo proposto por (MURALI; KASIF, 2003) procura identificar o maior, ou seja, aquele que contém o número máximo de genes conservados.

O algoritmo *xMotifs* (MURALI; KASIF, 2003) considera, como entrada, a matriz de expressão gênica e uma lista de intervalos que define os possíveis estados (ex: *down-regulated* e *up-regulated*) para cada gene. Para encontrar diferentes *biclusters* na matriz de dados, o seguinte procedimento iterativo é utilizado: encontrar o maior *xMotif* nos dados, remover as amostras que o atendem e, então, encontrar o maior *motif* nos dados restantes. Essa sequência é repetida até que todas as amostras satisfaçam à condição de algum *motif*. Essa sequência é repetida até que todas as amostras satisfaçam à condição de algum *motif*. Algumas das características do algoritmo *xMotifs* são:

- Permitir que um gene apareça em mais de um *motif* e em *motifs* que representam diferentes classes, modelando a possibilidade de

que o nível de expressão do gene possa ser regulado em várias condições;

- Se as amostras que correspondam a um *xMotif* computado não são removidas, também é possível que amostras apareçam em diferentes *motifs*;
- Não é necessário definir de antemão quantos *motifs* computar;
- Utilizando essa abordagem, é possível encontrar *xMotifs* com números muito diferentes de genes; o algoritmo tende a descobrir *motifs* com muitos genes nas primeiras iterações e *motifs* com menos genes nas iterações seguintes.

Além da matriz de expressão gênica, o algoritmo considera, para cada gene, uma lista de intervalos representando os estados nos quais um gene expresso numa amostra. O algoritmo assume que, para cada gene, os intervalos correspondentes aos seus estados são disjuntos. Ele amostra, aleatoriamente, de forma uniforme, n_s amostras, a partir do conjunto de dados original e os considera como sementes para encontrar *xMotifs*. Para cada semente, um conjunto de n_d conjuntos de amostras são selecionadas aleatoriamente, de forma uniforme. Cada conjunto possui s_d elementos, que são candidatos para inclusão no *motif*. Um *motif* é descartado se ele não possui uma fração mínima, α, das amostras. O algoritmo pode ser sumarizado nos seguintes passos:

1 Para i=1 até n_s fazer:

1.1 Escolher uma amostra c aleatoriamente;

1.2 Para j = 1 até n_d fazer:

1.2.1 Escolher um subconjunto D de amostras de tamanho s_d aleatoriamente

1.2.2 Para cada gene g, se g está no estado s em c em todas as amostras em D, incluir o par (g,c) no conjunto G_{ij}.

1.2.3 Seja C_{ij} = conjunto de amostras que concordam com c em todos os estados de genes em G_{ij}.

1.2.4 Descarte (C_{ij}, G_{ij}) se C_{ij} contém menos que αn amostras.

2 Retorne o *motif* (C*, G*) que maximiza . $|G_{ij}|$, $1 \leq i \leq n_s$, $1 \leq j < n_d$.

2.6 *Plaid Models*

Lazzeroni e Owen (2002) definem um modelo *plaid* como uma série de camadas aditivas que tentam capturar a estrutura subjacente de uma matriz de expressão gênica. O modelo inclui uma camada de fundo que contém todos os genes e as amostras, para explicar os efeitos globais dos dados. As camadas subsequentes representam efeitos adicionais, correspondentes aos *biclusters* que exibem um forte padrão não explicado pelo modelo geral.

No modelo *plaid*, o nível de expressão, a_{ij}, i = 1, . . . , n, j = 1, . . . , m, do i-ésimo gene na j-ésima amostra é modelada por:

$$a_{ij} = \Theta_{ij0} + \sum_{k=1}^{K} \Theta_{ijk}\rho_{ik}\kappa_{jk} + \varepsilon = (\mu + \alpha_{i0} + \beta_{j0}) + \sum_{k=1}^{K} (\mu_k + \alpha_{ik} + \beta_{jk})\ \rho_{ik}\kappa_{jk} + \varepsilon_{ij}$$

onde k é um índice de camada que começa no zero, para a camada de fundo, e vai até K, o número de *biclusters*; Θ_{ijk} é a soma da média (μ_k) e dos efeitos de gene e amostra na camada k (α_{ik} e β_{jk}, respectivamente); ρ_{ik} é um parâmetro binário de associação ao *bicluster* definido para cada k e igual a um se o gene i pertence ao k-ésimo *bicluster* e 0 em caso contrário; analogamente, κ_{jk} indica uma associação da amostra j ao *bicluster*; e ε_{ij} é o erro residual. O objetivo do algoritmo é encontrar um modelo que minimize o valor de:

$$Q = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} (a_{ij} - \Theta_{ij0} - \sum_{k=1}^{K} \Theta_{ijk}\rho_{ik}\kappa_{jk})^2$$

Para cada camada k, existem (2n−1)(2m−1) maneiras de selecionar os genes e as condições que participam do modelo. Por isso, não há garantiras de que o algoritmo encontre o melhor modelo para um determinado dado número, K, de camadas.

Supondo que as K-1 primeira camadas já tenham sido determinadas, para encontrar a K-ésima camada minimiza-se a seguinte expressão de Q:

$$Q = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} (Z_{ij} - \Theta_{ijk}\rho_{ik}\kappa_{jk})^2$$

onde $Z_{ij} = Z_{ij}^{k-1} = a_{lj} - \Theta_{ij0} - \sum_{k=1}^{K-1} \Theta_{ijk}\rho_{ik}\kappa_{jk}$ é o resíduo das primeiras K-1 camadas.

$\Theta^{(s)}$ denota todos os valores Θ_{ijK} na iteração s. De maneira semelhante, ρ(s) e $\kappa^{(s)}$ representam todos os valores ρ_{iK} e κ_{jK} na iteração s. Para s = 1, …, S, na iteração s, $\Theta^{(s)}$ é atualizado a partir de $\rho^{(s-1)}$ e $\kappa^{(s-1)}$, depois $\rho^{(s)}$ é atualizado a partir de $\Theta^{(s)}$ e $\kappa^{(s-1)}$, e, finalmente, $\kappa^{(s)}$ é atualizado a partir de $\Theta^{(s)}$ e $\rho^{(s-1)}$.

Assim, uma iteração do algoritmo consiste em atualizar o valor de Θ_{ij} dados os valores de ρ_i e κ_j, minimizando

$$Q = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}(Z_{lj} - (\mu_k+\alpha_i+\beta_j)\,\rho_i\kappa_j)^2$$

sujeito às condições

$$0 = \sum_{k=1}^{n}\rho_i^2\alpha_i = \sum_{j=1}^{m}\kappa_j^2\beta_j \quad .$$

A solução utilizando os multiplicadores de Lagrange é dada por:

$$\mu = \frac{\sum_i\sum_j \rho_i\kappa_j Z_{ij}}{\left(\sum_i \rho_i^2\right)\left(\sum_j \kappa_j^2\right)}$$

$$\alpha_i = \frac{\sum_j (Z_{ij} - \mu\rho_i\kappa_j)\kappa_j}{\rho_i\sum_j \kappa_j^2} \quad .$$

$$\beta_j = \frac{\sum_i (Z_{ij} - \mu\rho_i\kappa_j)\rho_i}{\kappa_j\sum_i \rho_i^2}$$

Dados os valores de Θ_{ij} e κ_j, o valor de ρ_i que minimiza Q é dado por:

$$\rho_i = \frac{\sum_j \Theta_{ij}\kappa_j Z_{ij}}{\sum_j \Theta_{ij}^2\kappa_j^2}$$

De forma equivalente, dados os valores de Θ_{ij} e ρ_i, o valor de κ_j que minimiza Q é dado por:

$$\kappa_j = \frac{\sum_j \Theta_{ij}\rho_j Z_{ij}}{\sum_i \Theta_{ij}^2 \rho_j^2}$$

A camada de fundo é a primeira a ser ajustada, sendo as demais camadas de *biclusters* adicionadas uma por vez até que um número pré-especificado é atingido ou camadas mais significativas não possam ser encontradas, como determinado por um teste de permutação.

Os valores de ρ_i e κ_j são inicializados com 0.5 + pequeno valor aleatório. O algoritmo permite que ρi e κ_j assumam valores reais (não binários), sendo que apenas nas últimas iterações é que o algoritmo impõe a restrição de que as variáveis ρ_i e κ_j assumam valores binários.

2.7 *Bicluster* espectral

A abordagem de (KLUGER et al., 2003), denominada *bicluster* espectral, assume que a matriz de dados contém uma estrutura de tabuleiro após um processo de normalização. Supondo que a matriz original é A e a resultante é a matriz normalizada A', a ideia é resolver o problema de autovalores $A'^{T}A' x = \lambda^2 x$ e examinar os autovetores x. Se as constantes em um autovetor podem ser ordenadas para produzir uma estrutura em degraus, os *clusters* de colunas podem ser identificados. Os *clusters* de linhas são encontrados de forma semelhante, a partir de y satisfazendo $A'^{T}A' y = \lambda^2 y$. Mais precisamente, é possível mostrar que o padrão de tabuleiro em uma matriz é refletido nas estruturas constantes do par de vetores x e y que resolvem o problema de autovalor $A'^{T}A' x = \lambda^2 x$ e $A'TA y = \lambda^2 y$, onde x e y possuem autovalores comuns.

O algoritmo depende criticamente do processo de normalização utilizado para transformar a matriz A. Existem três métodos de normalização. O primeiro método (redimensionamento independente de linhas e colunas) pressupõe que a matriz não-normalizada é obtida multiplicando-se cada linha i por um escalar r_i e cada coluna j por um escalar c_j , então,

$$\frac{r_{i1}}{r_{i2}} = \frac{\textit{média da linhai1}}{\textit{média da linhai2}} = \frac{a_{i1J}}{a_{i2J}} \quad .$$

Assumindo que R é uma matriz diagonal com entradas r_i na diagonal e C é uma matriz diagonal definida de forma semelhante, então o problema de autovetores pode ser formulado pelo redimensionamento da matriz de dados: $\hat{A} \equiv R^{-1/2} AC^{-1/2}$. O segundo método (bi-estocastização) funciona pela repetição da padronização independente de linhas e colunas até que a estabilidade seja alcançada. Na matriz final, tanto a soma de todas as linhas quanto a soma de todas as colunas são constantes, embora de valores diferentes. Finalmente, o terceiro método (log-iterações) assume que, se as linhas/colunas originais diferem por constantes multiplicativas, então, após tomar seu logaritmo, elas diferem por constantes aditivas. Além disso, cada linha e coluna deve ter média igual a zero. Isto pode ser conseguido por meio da transformação de cada entrada da seguinte forma: $a_{ij}' = a_{iJ} - a_{Ij} - a_{Ij} + a_{IJ}$. Note que a_{ij}' é o resíduo de cada elemento da matriz a_{ij} da matriz de dados A, tal como definido pela equação (8) na seção 5.1.

A matriz $\hat{A}$ apresentará os *biclusters* com estrutura de tabuleiro.

2.8 Classificação

Diferentes abordagens heurísticas têm sido aplicados ao problema de encontrar *biclusters* em matrizes de dados. Madeira et al. (2004) as classificam em cinco classes:

- Combinação de clustering de linhas e colunas iterativamente;
- Dividir e conquistar;
- Busca iterativa gulosa;
- Enumeração exaustiva de *bicluster*;
- Identificação da distribuição de parâmetros que descrevem *biclusters*.

A Tabela 1 apresenta uma comparação global dos algoritmos apresentados, classificando-os de acordo com o tipo e estrutura de *bicluster* considerado e a abordagem heurística utilizada:

Tabela 1. Classificação de algoritmos de biclustering.

Algoritmo	Tipo	Estrutura	Abordagem
CC	Valores coerentes	Sobrepostos e posicionados arbitrariamente	Busca iterativa gulosa
FLOC	Valores coerentes	Sobrepostos e posicionados arbitrariamente	Busca iterativa gulosa
ISA	Valores coerentes	Sobrepostos e posicionados arbitrariamente	Busca iterativa gulosa
Bimax	Valores coerentes	Sobrepostos e posicionados árvore	Dividrir e conquistar
xMotis	Evoluções coerentes	Único ou sobrepostos e posicionados arbitrariamente	Busca iterativa gulosa
Plaid Models	Valores coerentes	Sobrepostos e posicionados arbitrariamente	Identificação da distribuição de parâmetros
Spectral	Valores coerentes	Não sobrepostos e com estrutura de tabuleiro	Busca iterativa gulosa

3. Aplicações

Todos os algoritmos apresentados na seção 2 estão implementados nos pacotes biclust, bicARE e ISA do software de análise estatística R (R DEVELOPMENT CORE TEAM, 2010). Nessa seção, dados de expressão gênica de extremos de resistência a carrapatos (IBELLI et al., 2010), coletados no escopo do projeto Rede Genômica Animal são utilizados para ilustrar a análise de *biclustering* para análise de dados de microarranjos.

Em cada um dos próximos itens, as ilustrações correspondem às visualizações dos resultados, comandos executados no console R e as correspondentes saídas.

3.1 *Script* para o algoritmo CC

1. Comandos no console R para execução do algoritmo de Cheng e Church (2000):

```
> data <- as.matrix(read.table("dfRMA_LNt.csv",
header=TRUE, sep=";", dec=",",
+ row.names=1))
> cc <- biclust(data, method=BCCC(), delta=1.5, al-
pha=1, number=10)
> cc
```

2. Saída no console R:

```
An object of class Biclust

call:
    biclust(x = data, method = BCCC(), delta = 1.5, al-
pha = 1, number = 10)

Number of Clusters found:  10
```

```
First 5 Cluster sizes:
          BC 1   BC 2   BC 3   BC 4   BC 5
Number of Rows:    "1229" "   9" "   8" "   9" "   8"
Number of Columns: "  36" "   9" "   8" "   9" "   8"
```

3. Comando R para criação do gráfico apresentado na Figura 4:

```
> drawHeatmap(data, bicResult=cc, number=4, local=TRUE)
```

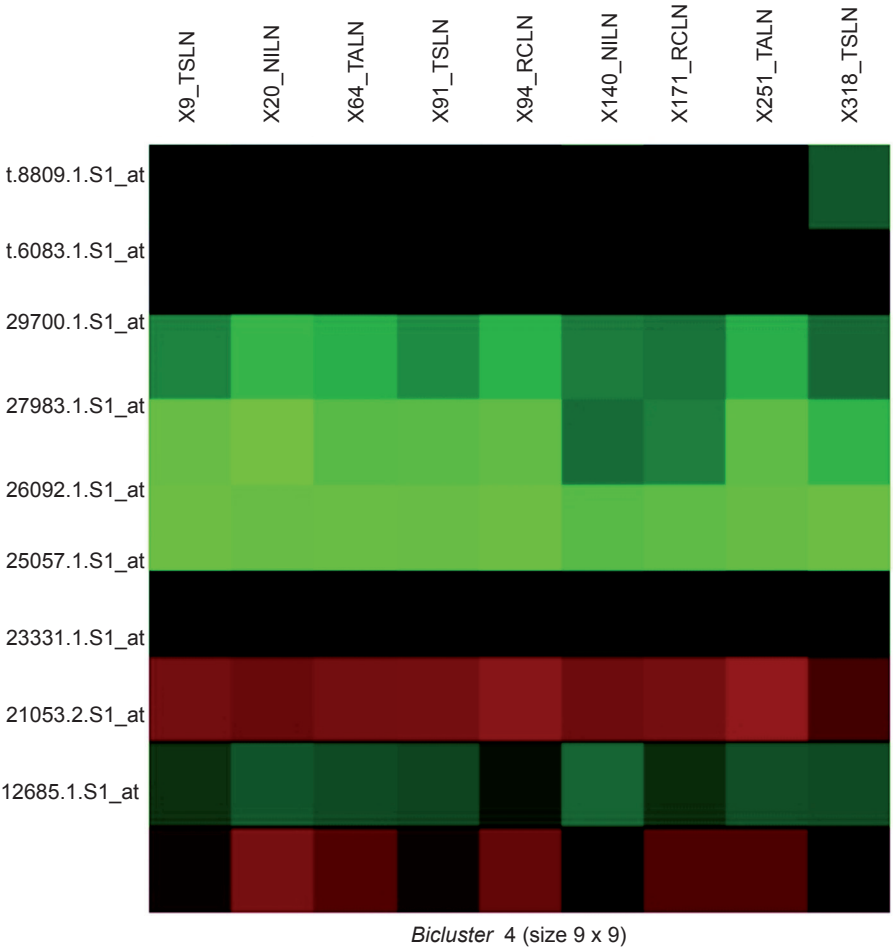


Figura 4. *Heatmap* para *bicluster* identificado por CC.

3.2 *Script* para o algoritmo FLOC

1. Comandos no console R para execução do algoritmo FLOC (YANG et al., 2005):

```
> data <- readExpressionSet("dfRMA_LNt.csv",
header=TRUE, sep=";", dec=",",
+ row.names=1)
> floc <- FLOC(data, k=20, pGene=0.5, pSample=0.5,
r=0.01, 5, 5, 100)
> floc
```

2. Saída no console R:

```
Call :
FLOC(Data = data, k = 20, pGene = 0.5, pSample = 0.5, r
= 0.01,
    N = 5, M = 5, t = 100)


Parameters :
 number of biclusters :  20
 residu threshold :  0.01
 gene initial probability :  0.5
 sample initial probability :  0.5
 number of iterations :  100
 date :  Thu Dec 09 17:27:45 2010
Biclusters :
      residue volume genes conditions     rowvar
 [1,] 0.011145080   3248   464          7 0.01354530
 [2,] 0.011191233   3519   391          9 0.01423178
 [3,] 0.009753903   3190   638          5 0.01368312
 [4,] 0.012365532   4464   248         18 0.01889194
 [5,] 0.009994406   3030   606          5 0.01354449
 [6,] 0.011531173   3150   350          9 0.01369026
 [7,] 0.009987370   3234   539          6 0.01298075
 [8,] 0.009990257   3025   605          5 0.01291756
```

```
 [9,] 0.011765769   4068   226          18 0.02206667
[10,] 0.010859357   2709   301           9 0.01306471
[11,] 0.009979462   2980   596           5 0.01421696
[12,] 0.010078844   2730   546           5 0.01335761
[13,] 0.009988003   3275   655           5 0.01375893
[14,] 0.010008762   2825   565           5 0.01265620
[15,] 0.009975368   3408   568           6 0.01211533
[16,] 0.010410743   3220   460           7 0.01320009
[17,] 0.010170981   2725   545           5 0.01315352
[18,] 0.009997949   2985   597           5 0.01319190
[19,] 0.010345552   2560   512           5 0.01371991
[20,] 0.009977722   2880   576           5 0.01279177
```

3. Comandos R para criação do gráfico apresentado na Figura 5:

```
> bic <- bicluster(floc, 6, graph=FALSE)
> plot(bic)
```

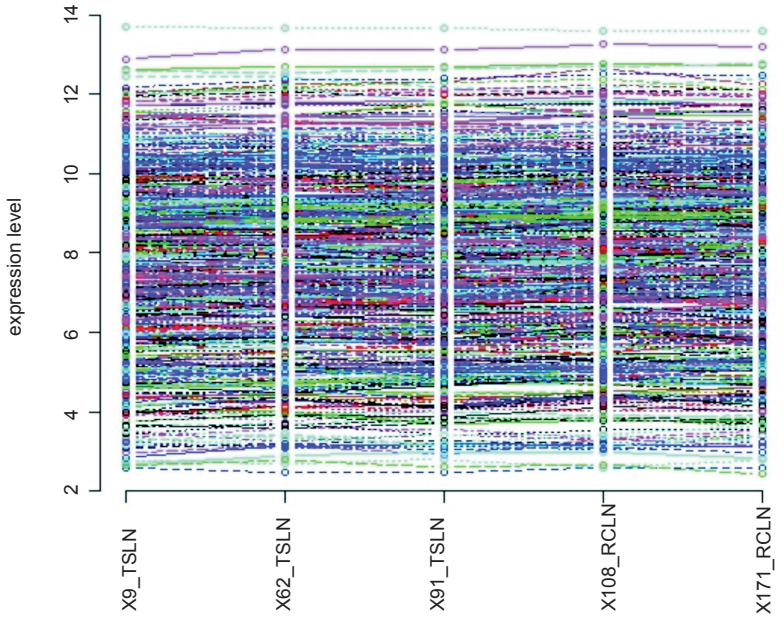


Figura 5. Nível de expressão para *bicluster* identificado por FLOC.

3.3 *Script* para o algoritmo ISA

1. Comandos no console R para execução do algoritmo ISA (BERGMANN et al., 2003):

```
> data <- readExpressionSet("dfRMA_LNt.csv", annotation="bovine", he-
ader = TRUE,
+ sep=";", dec=",", row.names=1)
> modules <- ISA(data, thr.gene=1, thr.cond=1)
> modules
> getFeatureScores(modules, 1)
> getSampleScores(modules, 1)
```

2. Saída no console R:

```
Loading required package: org.Bt.eg.db


An ISAModules instance.
  Number of modules: 1
  Number of features: 185
  Number of samples: 36
  Gene threshold(s): 1
  Conditions threshold(s): 1

[[1]]
Bt.28162.2.A1_at  Bt.29332.1.A1_at  Bt.12140.1.S1_at  Bt.270.1.S1_at
   0.8324656       -0.9562532         -0.7204452       -0.8569374
Bt.13053.1.S1_at  Bt.5203.1.S1_at  Bt.4679.1.S1_at  Bt.12301.1.S1_at
   -0.9833823       -0.8765244          0.8341574       -0.7240979
Bt.5528.1.S1_at   Bt.8686.1.S1_at  Bt.211.1.S1_at   Bt.28214.1.S1_at
   -0.8971297       -0.7766515         -0.9117565       -0.7598037
Bt.543.1.S1_at  Bt.3856.1.S1_at   Bt.27008.1.A1_at  Bt.27983.1.S1_at
   0.7823459        -0.9828176          0.7992970       -0.8532411
Bt.5080.1.A1_at  Bt.29166.1.S1_at  Bt.23128.3.S1_at  Bt.29939.2.S1_at
   -0.9421002       -0.8009950         -0.9605986       -0.7989219
```

```
Bt.20230.1.S1_at  Bt.21800.1.S1_at  Bt.19482.1.A1_at  Bt.20472.1.S1_at
   0.8215330         -0.7138544       -0.9017482       -0.9467809
Bt.13628.2.S1_a_at  Bt.19999.2.S1_at  Bt.6840.1.S1_a_at Bt.27854.1.S1_at
   -0.7928793        -0.8678991       -0.8948464       -0.7306115
Bt.10885.2.S1_at  Bt.27237.1.S1_a_at  Bt.22664.1.A1_at  Bt.22275.1.A1_at
   -0.8287618         -0.8172985       0.7929191       -0.9627211
Bt.12697.1.S1_a_at  Bt.5365.1.S1_at  Bt.7319.1.S1_at  Bt.1199.1.S1_at
   0.7691601          0.8867160        0.8309938       -0.7908283
Bt.22010.1.S1_at  Bt.27144.1.S1_at  Bt.20356.1.S1_at  Bt.23351.1.S1_at
   -0.8778380         0.7729109       -0.8619260        -0.9109961
Bt.22199.1.S1_at  Bt.26770.1.S1_at  Bt.2806.1.S1_at  Bt.16150.1.A1_at
   0.7718338         -0.8595905       0.8042903        -0.7413135
Bt.4175.2.S1_at  Bt.8496.1.S1_at  Bt.27079.1.S1_at    Bt.1740.1.S1_at
   -0.9191478        -0.8884781       -0.8689247        -0.8753742
Bt.27167.1.A1_at  Bt.3071.1.S1_at  Bt.24676.1.S1_at  Bt.20118.1.S1_at
   -0.8390571        -0.8056264       -0.8789137        -0.8933528
Bt.24513.1.S1_at   Bt.9269.1.S1_at  Bt.15927.1.S1_at  Bt.8356.1.S1_at
   -0.8590008        -0.9558564       -0.7736253         -0.7287719
Bt.910.2.S1_at   Bt.7439.1.S1_at  Bt.13588.3.A1_at   Bt.22527.1.A1_at
   -0.8020878        -0.7736223       -0.8209881         -0.8136981
Bt.8829.1.S1_a_at  Bt.19064.2.S1_at  Bt.5711.2.S1_at  Bt.3448.3.S1_at
   0.8283641        -0.7681912         -0.8078682         -0.7597359
Bt.8809.1.S1_at  Bt.21482.2.S1_at  Bt.20846.1.A1_at  Bt.18808.1.S1_at
   0.9435644        -0.8862078          0.8062575          0.8846014
Bt.24899.1.A1_at  Bt.3520.1.S1_at  Bt.27091.1.S1_at Bt.24477.1.S1_a_at
   0.7675187        -0.9618288          0.7904736         -0.7998438
Bt.26375.1.A1_at  Bt.10236.1.S1_at  Bt.24779.2.S1_at  Bt.23611.2.S1_at
   -0.7538189       -0.8823660         -1.0000000         -0.7069021
Bt.21056.1.S1_at  Bt.26818.1.S1_at   Bt.7078.1.S1_at   Bt.29956.1.A1_at
   -0.8088560       -0.7402126         -0.8116901          0.8531650
Bt.26293.1.A1_at  Bt.19823.1.A1_at   Bt.24023.1.A1_at Bt.13553.2.S1_a_at
-0.7401579          -0.8081429          0.7740064         -0.8777225
```

```
Bt.20895.1.S1_at   Bt.23516.1.S1_at   Bt.10039.1.S1_at
   -0.9014634         -0.7737387        -0.7684132

[[1]]
X9_TSLN    X55_TALN    X58_NILN    X94_RCLN   X101_TALN   X102_NILN
0.04387471  0.09162413  0.06630616  0.12936957 -0.83009763 0.16103934
X112_NILN  X125_RCLN   X140_NILN   X153_TSLN   X163_NILN   X173_TSLN
0.18975345 -0.74007696 -0.98402764 -1.00000000 0.08045014 -0.72248338
X206_RCLN    X208_NILN   X251_TALN    X261_RCLN    X318_TSLN
-0.73648186  0.04435126  0.15829995  -0.74164561   -0.82572623
```

3. Comandos R para criação do gráfico apresentado na Figura 6:

```
> ISA2heatmap(modules, 1, data, norm = "sample", scale
= "none", col=RGBColVec(64))
```

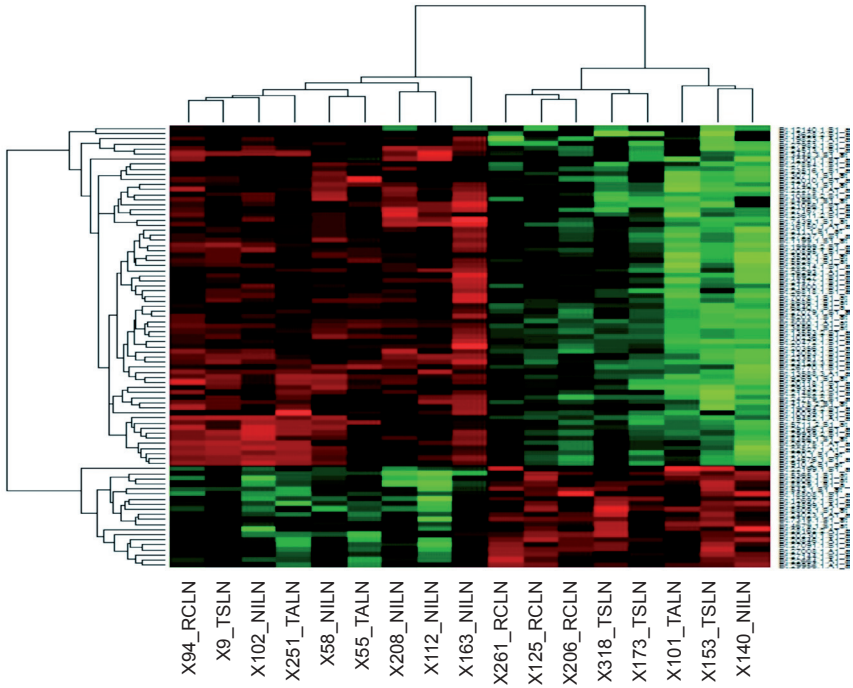


Figura 6. *Heatmap* para *bicluster* identificado utilizando ISA.

3.4 *Script* para o algoritmo Bimax

1. Comandos no console R para execução do algoritmo BIMAX (PRELIC et al.,2006):

```
> data <- binarize(as.matrix(read.table("dfRMA_LNt.
csv", header=TRUE, sep=";",
+ dec=",", row.names=1)))
> bi <- biclust(x=data, method=BCBimax(), minr=5,
minc=5, number=20)
> bi
```

2. Saída no console R:

```
[1] "Threshold:  8.26289607"

An object of class Biclust

call:
    biclust(x = data, method = BCBimax(), minr = 5,
minc = 5, number = 20)

Number of Clusters found:  20

First  5  Cluster sizes:
                  BC 1  BC 2  BC 3  BC 4  BC 5
Number of Rows:    "448" "451" "438" "441" "437"
Number of Columns: "  5" "  5" "  5" "  5" "  6"
```

3. Comandos R para criação do gráfico apresentado na Figura 7:

```
> drawHeatmap(data, bicResult=bi, number=5, local=TRUE)
```

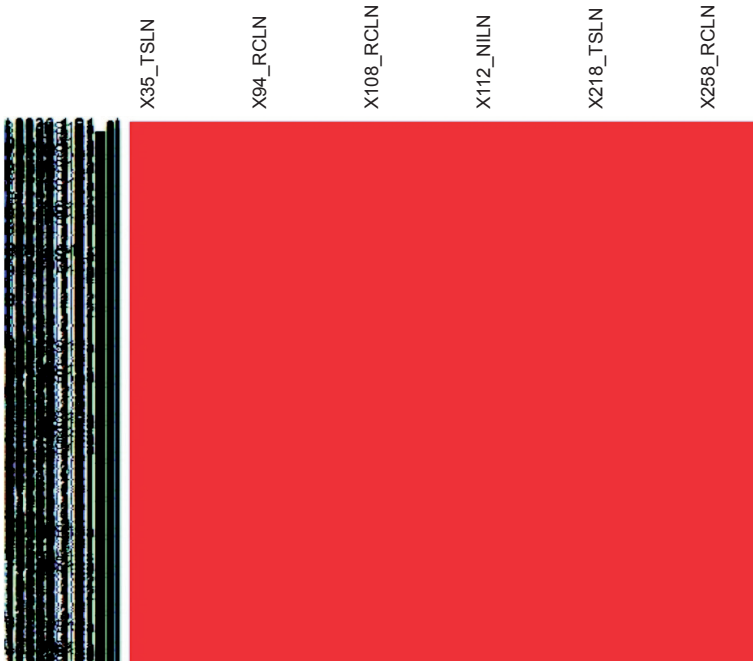


Figura 7. *Heatmap* para *bicluster* identificado utilizando Bimax.

3.5 Script para o algoritmo *xMotifs*

1. Comandos no console R para execução do algoritmo *xMotif* (MURALI.; KASIF, 2003):

```
> data <- discretize(as.matrix(read.table("dfRMA_LNt.
csv", header=TRUE, sep=";",
+ dec=",", row.names=1)))
> motif <- biclust(data, method=BCXmotifs(), ns=20,
nd=20, sd=5, alpha=0.01
+ number=10)
> motif
```

2. Saída no console R:

```
An object of class Biclust
```

```
call:
        biclust(x = data, method = BCXmotifs(), ns =
20, nd = 20, sd = 5,
            alpha = 0.01, number = 10)


Number of Clusters found:  10


First  5  Cluster sizes:
                  BC 1  BC 2  BC 3  BC 4  BC 5
Number of Rows:    "656" "286" " 85" " 54" " 23"
Number of Columns: "  6" "  6" "  6" "  6" "  6"
```

3. Comandos R para criação do gráfico apresentado na Figura 8:

```
> drawHeatmap(data, bicResult=motif, number=5,
local=TRUE)
```

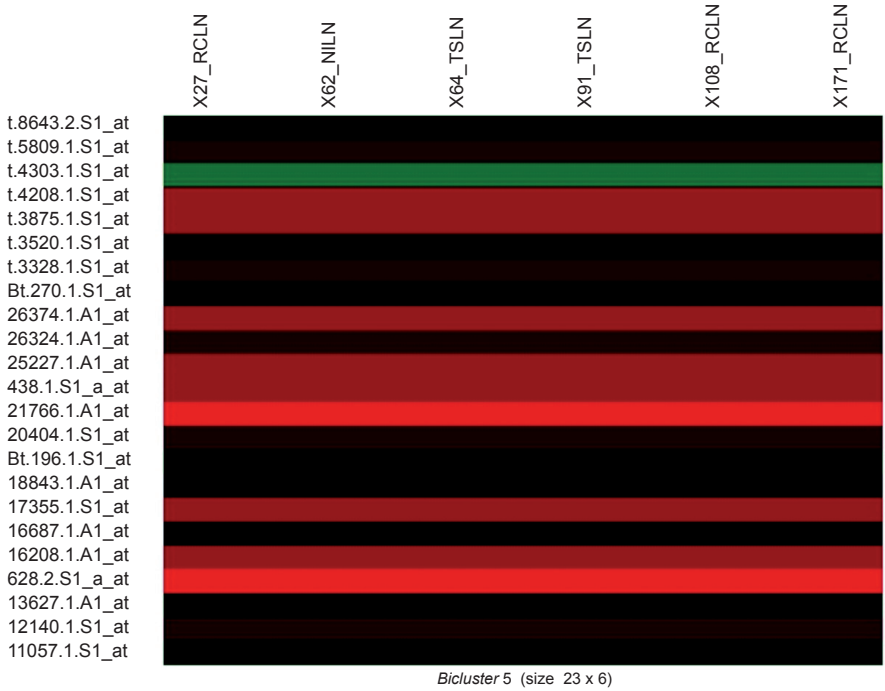


Figura 8. *Heatmap* para *bicluster* identificado utilizando *xMotifs*.

3.6 *Script* para *Plaid Models*

1. Comandos no console R para execução do modelo *Plaid* (LAZZERONI; OWEN, 2002):

```
> data <- as.matrix(read.table("dfRMA_LNt.csv",
header=TRUE, sep=";", dec=",",
+ row.names=1))
> plaid <- biclust(data, method=BCPlaid())
> plaid
```

2. Saída no console R:

```
An object of class Biclust

call:
    biclust(x = data, method = BCPlaid())

Number of Clusters found:  8

First  5  Cluster sizes:
                  BC 1  BC 2  BC 3  BC 4  BC 5
Number of Rows:    "159" " 57" " 26" " 88" " 67"
Number of Columns: "  9" "  9" "  9" "  7" "  7"
```

3. Comandos R para criação do gráfico apresentado na Figura 9:

```
> drawHeatmap(data, bicResult=plaid, number=8,
local=TRUE)
```

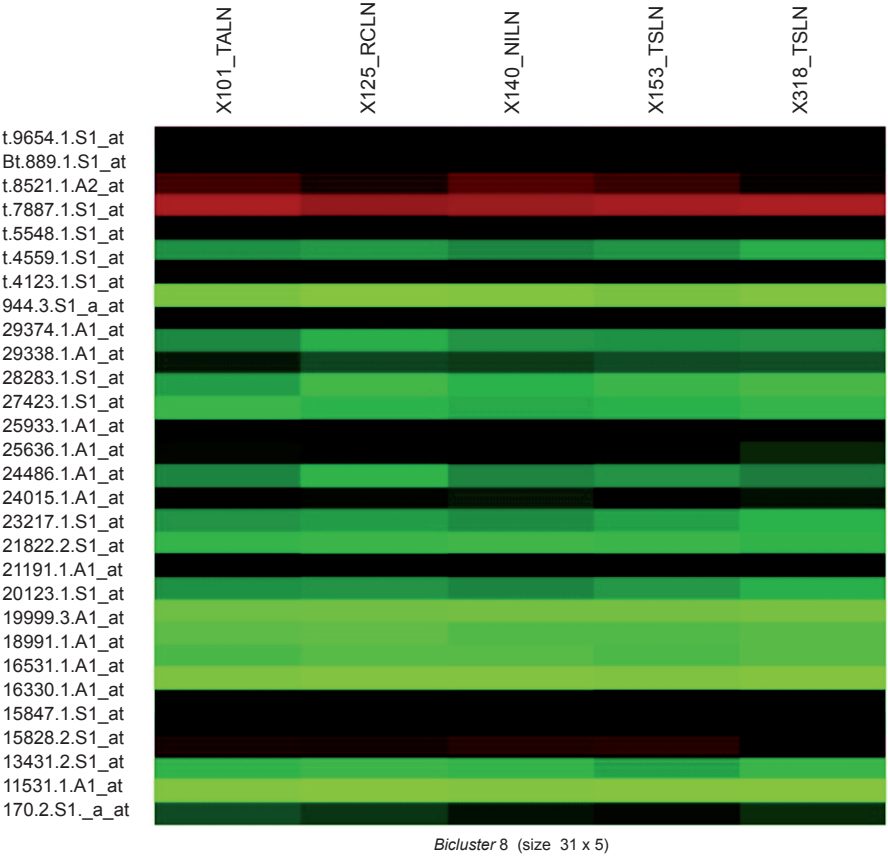


Figura 9. *Heatmap* para *bicluster* identificado utilizando *Plaid Models*.

3.7 Script para Spectral

1. Comandos no console R para execução do modelo Spectral (KLUGER et al., 2003):

```
> data <- as.matrix(read.table("dfRMA_LNt.csv",
header=TRUE, sep=";", dec=",",
+ row.names=1))
> spec <- biclust(data, method=BCSpectral(), numberO-
```

```
fEigenvalues=20)
```

2. Saída no console R:

```
Warning message:
In spectral(x, normalization, numberOfEigenvalues,
minr, minc, withinVar) :
  No biclusters found
```

Uma vez que o tipo de *bicluster* considerado por esse algoritmo é o não sobreposto e com estrutura de tabuleiro, o fato de o algoritmo não ter sido capaz de identificar nenhum *bicluster* parece ser uma evidência de que os dados analisados não apresentam essa estrutura.

4 Conclusão

Neste trabalho, foi apresentada uma revisão de alguns algoritmos de *biclustering* com foco de aplicação em análise de dados de expressão gênica utilizando a tecnologia de microarranjos. Os métodos estudados são aqueles implementados nos pacotes *biclust*, *bicARE* e ISA do software de análise estatística R (R DEVELOPMENT CORE TEAM, 2010).

Embora apenas um conjunto de dados tenha sido utilizado para testar os métodos implementados no R, os algoritmos Bimax e Spectral parecem ser os menos indicados para análise de dados expressão gênica. O primeiro considera apenas valores diretos (0 ou 1), que levar á perda de informação, equanto que Spectral exige que os *biclusters* não se sobreponham e apresentem uma estrutura de tabuleiro, o que raramente acontece com dados reais. Não por acaso, nenhum *bicluster* foi encontrado no conjunto de dados testado.

Muitos algoritmos são parametrizados e, por isso, diferentes valores de parâmetros devem ser testados. Por exemplo, no ISA, quanto menores os valores dos parâmetros limitantes, menos restrito será o critério utilizado para encontrar *biclusters*. Várias combinações de valores foram testados, mas apenas um *bicluster* foi encontrado. Questões específicas da implementação também devem ser considerados como, por exemplo, no caso do FLOC que, ao contrário das demais implementações, apresenta o *bi-*

cluster encontrado utilizando um gráfico que não é um heatmap e, portanto, deve ser interpretado de forma distinta.

De maneira geral, ao realizar uma análise, é preciso calibrar o algoritmo que melhor se adeque à estrutura dos dados sendo analisados. Deve-se experimentar diferentes algoritmos, bem como diferentes valores de parâmetros e sempre ter em mente a estrutura de *bicluster* focada pelo método.

5 Referências

BERGMANN, S.; IHMELS, J.; BARKAI, N. Iterative signature algorithm for the analysis of large-scale gene expression data. **Physical Review**, v. 67, p. 1-18, 2003.

CHENG, Y.; CHURCH, G. M. Biclustering of expression data. In: INTERNATIONAL CONFERENCE ON INTELLIGENT SYSTEMS FOR MOLECULAR BIOLOGY, 8., Toronto. **Proceedings**... California: AAAI Press , 2000. p. 93-103.

IBELLI, A. M. G.; HIGA, R. H.; GIACHETTO, P. F.; YAMAGISHI, M. E. B.; OLIVEIRA, M. C. S.; CARDOSO, F. F.; ALENCAR, M. M.; REGITANO, L. C. A. **Genes e vias metabólicas envolvidos nos mecanismos de resistência e susceptibilidade de bovinos infestados com carrapato Rhipicephalus microplus.** In: CONGRESSO BRASILEIRO DE GENÉTICA, 56., 2010, Guarujá. Resumos... Ribeirão Preto: Sociedade Brasileira de Genética, 2010. p. 74.

KLUGER, Y.; BASRI, R.; CHANG, J. T.; GERSTEIN, M. Spectral Biclustering of Microarray Data: Coclustering Genes and Conditions, Genome Research, v. 13, p. 703-716, 2003.

LAZZERONI, L.; OWEN, A. Plaid models for gene expression data. **Statistica Sinica**, v. 12, p. 61-86, 2002.

MADEIRA, S. C; OLIVEIRA, A. L. Biclustering algorithms for biological data analysis: a survey, **IEEE Transactions on Computational Biology and Bioinformatics**, v.1, n.1, p.24-45, 2004.

MURALI, T.; KASIF, S. Extracting conserved gene expression *motifs* from gene expression data. In: PACIFIC SYMPOSIUM ON BIOCOMPUTING,

2002, Lihue. **Proceedings**... Singapore: World Scientific, 2002. p. 77-88.

PRELIC, A.; BLEULER, S.; ZIMMERMANN, P.; WIL, A.; BUHLMANN, P.; GRUISSEM, W.; HENNIG, L.; THIELE, L. ; ZITZLER, E. A systematic comparison and evaluation of *biclustering* methods for gene expression data. **Bioinformatics**, v. 22, n. 9, p. 1122-1129, 2006.

R DEVELOPMENT CORE TEAM. **R**: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. 2010. Disponível em: <http://www.R-project.org>. Acesso em: 13 out. 2010.

YANG, J.; WANG, H., WANG, W.; YU, P. An improved *biclustering* method for analyzing gene expression. **International Journal on Artificial Tools**, v. 14, n. 5, p. 771–789, 2005.

Informática Agropecuária
Pecuária Sudeste

Ministério da
Agricultura, Pecuária
e Abastecimento

CGPE 9146